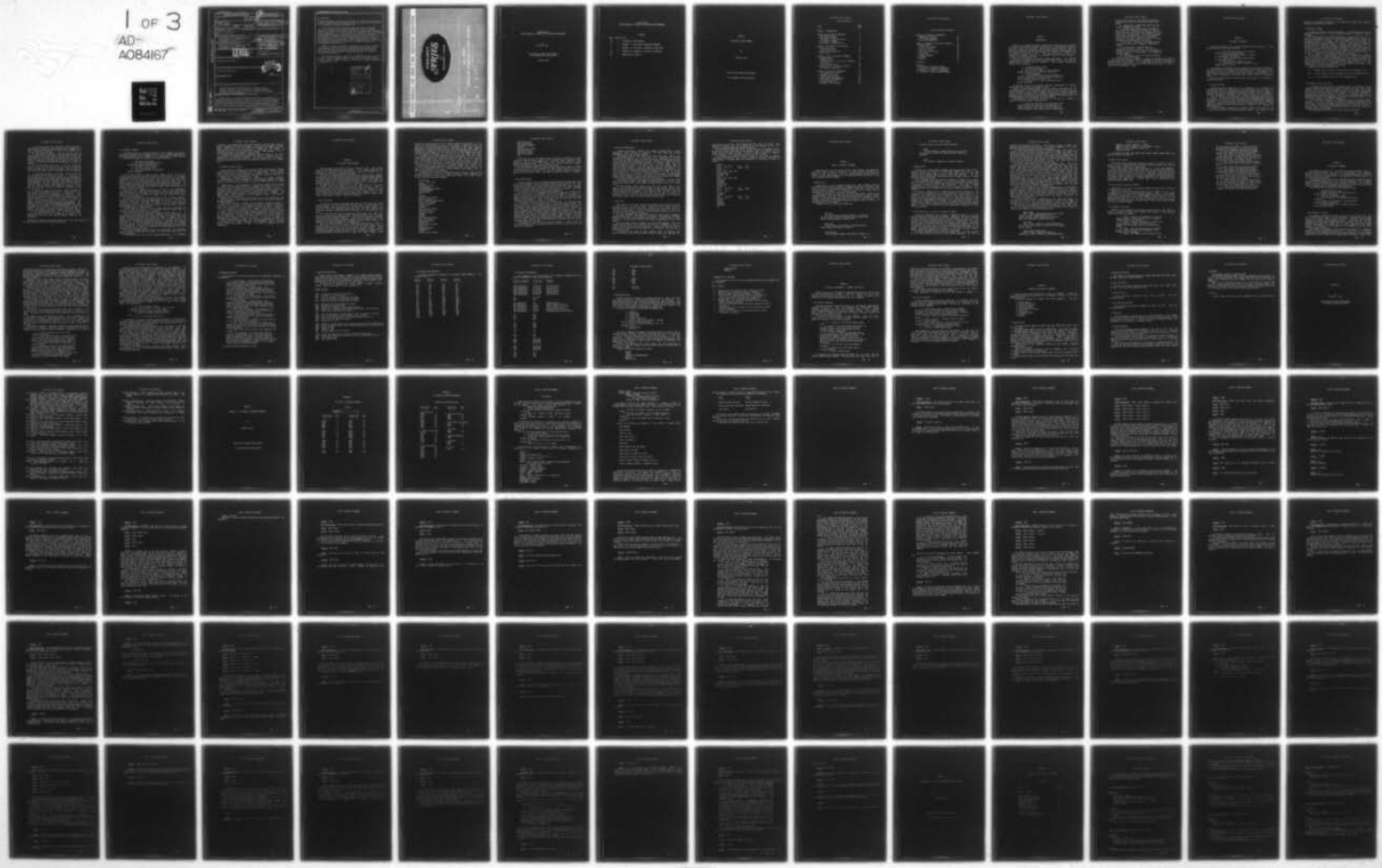
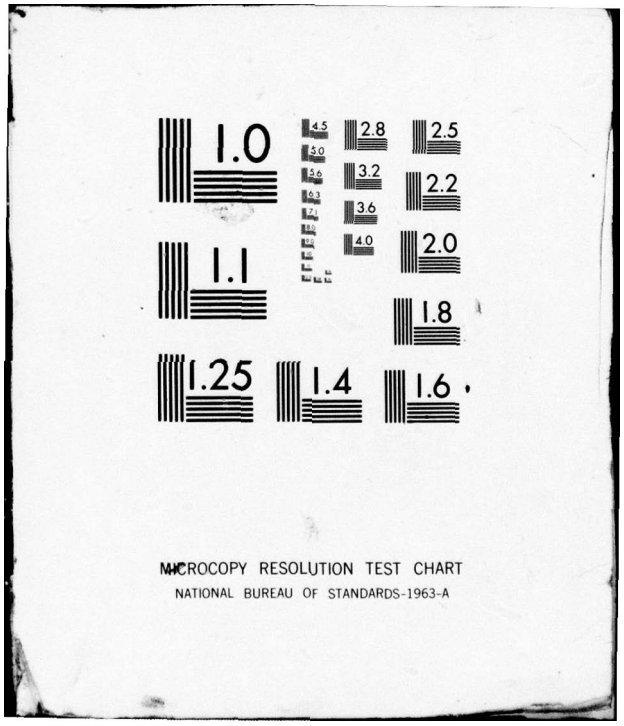
AD-A084 167 ARMY SATELLITE COMMUNICATIONS AGENCY FORT MONMOUTH NJ F/G 9/2
PROJECT ARIES. USER'S MANUALS FOR ARIAN II AND ASSOCIATED SUBSY--ETC(U)
APR 80 R L CONN

UNCLASSIFIED NL

1 OF 3
AD-
A084167




1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A084167
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) PROJECT ARIES. USER'S MANUALS for ARIAN II and ASSOCIATED SUBSYSTEMS.	5. TYPE OF REPORT & PERIOD COVERED Final rept. May 79-May 80, 5/79 to 5/80
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard L. Conn	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Satellite Communications Agency USA CORADCOM, Attn: DRCPM-SC-4G Ft Monmouth, NJ 07703	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Elt: 6.11.01.A Proj: 1L1 61101 A91A Task: 33 Work Unit: 131
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Satellite Communications Agency USA CORADCOM, Attn: DRCPM-SC-4G Ft Monmouth, NJ 07703	12. REPORT DATE 1 May 1980 (11) 18 Apr 80
	13. NUMBER OF PAGES ~~283~~ (12) 269
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) LEVEL II	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

DTIC ELECTE S MAY 9 1980 D C

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Distribution Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

8080 Microprocessor, Z80 Microprocessor, Operating System, Software Development System, Interactive Software Development, Assembly Language, Editor, Floppy Disk, Interactive Environment, Microprocessor, Microcomputer

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Project ARIES is a research and development effort whose objective is to investigate and analyze as well as develop an interactive assembly language software development technique and system. Tests comparing the development capability and speed of conventional assembly language programming techniques (cross assembly, resident assembly via a "conventional" operating system, and cross compilation) and an early implementation of the proposed technique

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20 (continued)

(software development system) have shown that this technique of interactive assembly language software development promises to provide a factor of from 3 to 10 decrease in program development time.

Project ARIES, which was started by the author while in graduate school, has resulted in the creation of ARIAN II, a floppy disk-based operating and software development system for interactive assembly language software development. It is an integrated tool designed to work specifically with the ARIES-I hardware configuration, and its 34 resident commands give the user the abilities to create and manipulate text and binary files, to assemble the text of an assembly language source program, to execute and breakpoint such a program with a number of debugging aids, to communicate with an external computer via a modem, and to examine and modify the microcomputer's memory directly.

ARIAN II's resident commands are supplemented by disk-based transient commands which may be created at the discretion of the user and loaded into the microcomputer and executed. The user also has the ability to create temporary memory-resident commands which may also be invoked by the user at his discretion.

The software development capabilities of ARIAN II are notably limited to "small" assembly language programs on the order of 40K bytes (source) or less. ARIAN II, however, provides a useful tool for software development of programs of this size.

USER'S MANUALS FOR ARIAN II AND ASSOCIATED SUBSYSTEMS

PROJECT ARIES

by

RICHARD L. CONN

SATCOM

U.S. ARMY SATELLITE COMMUNICATIONS AGENCY

U.S. ARMY

SATELLITE COMMUNICATIONS AGENCY

FORT MONMOUTH, NEW JERSEY

PROJECT ARIES
USER'S MANUALS for ARIAN II and ASSOCIATED SUBSYSTEMS

by
Richard L Conn

USA Satellite Communications Agency
Fort Monmouth, New Jersey 07703

18 April 1980

Project ARIES
USER'S MANUALS for ARIAN II and ASSOCIATED SUBSYSTEMS

Contents

Section 1

The ARIAN II USER'S MANUAL

by

Richard L Conn

USA Satellite Communications Agency

Fort Monmouth, New Jersey 07703

The ARIAN II User's Manual

Table of Contents

Table of Contents, Continued

The ARIAN II User's Manual

CHAPTER 1

INTRODUCTION

ARIAN II is an operating system, a program which allows its user to execute and control other programs, designed specifically for a microcomputer possessing the ARIES-I hardware configuration. Based on the original ARIAN, or ARIAN I, ARIAN II is extensively integrated with the ARIES-I hardware configuration; the microcomputer hardware and the ARIAN software are designed to work together with each other's configuration in mind. This type of design -- that of an integration of hardware, firmware, and software within a single system -- produces a unique and useful tool.

This tool is designed to be used for software development. The hardware configuration was organized and coordinated with ARIAN to give the user the maximum amount of benefit from his resources. This hardware configuration consists of the following:

1. at least 40K bytes of RAM,
2. the Z80 microprocessor,
3. two 5 1/4-inch floppy disk drives (Shugart SA-400) with North Star floppy disk controller,
4. a PROM programmer,
5. a high-speed (9600 baud) CRT and keyboard, a printer, a modem, and a second high-speed CRT VDM, with their associated serial I/O ports, and
6. three parallel I/O ports, including a front panel LED display.

As the user can see, the ARIES-I hardware configuration is designed specifically with software development in mind.

ARIAN II is specifically designed to aid in software development for the Z80 microprocessor. Its 34 commands give the user the abilities to create and manipulate text and binary files, to assemble the text of an assembly language file, to execute and breakpoint his assembly language programs with a number of debugging aids, to communicate with an external computer via the modem, and to examine and modify the microcomputer's memory directly.

While ARIAN II is running, the user may be in any one of a number of command or data entry modes. These mode are:

1. Block Line Entry Mode. This mode permits the entry of a block of lines into the primary file. It allows the user to enter lines into the primary file without typing the line numbers; ARIAN II automatically

prefixes each line with a line number and optionally renumbers the primary file when the user exits this mode.

2. Command Mode. Command mode permits the user to type a command to ARIAN II.

3. Display Mode. Display mode is the mode in which ARIAN II is displaying a directory or lines of text to the user. He can stop this display while it is being created by keying <ESC>. All displays are paged, and the user is prompted with a "?" at the bottom of each page to find out if he wishes to continue the display; the user must respond with "N" or "n" to stop the display and anything else to continue.

4. Edit Mode. This mode is the command system invoked by the EDIT Level 2 command or XEDIT Level 3 command.

5. Terminal Mode. Terminal mode is an interactive subsystem of ARIAN II. The user can communicate with an external computer via the modem and acoustic coupler while in this mode. It is invoked by the TERM command.

6. Utility Mode. This is the Utility Subsystem, invoked by the UTIL command.

This is a user's manual for ARIAN II. ARIAN II is different from ARIAN I in many respects, and the ARIAN I user is encouraged to read this manual and its appendices before using ARIAN II. It is hoped that the user will find ARIAN II to be at least as useful and as far above its predecessor, ARIAN I, as ARIAN I was above its predecessors. IDIC

CHAPTER 2

The ARIAN II EXECUTIVE

The heart of ARIAN II is a very small program called the Executive. This program performs the following functions:

1. Resets the system stack, sets interrupt mode zero, and disables interrupts,
2. Displays the ARIAN II prompt to the user via the principal I/O channel,
3. Inputs a command from the user through the input line editor,
4. Parses the input command line, and
5. Searches for and transfers control to the command routine if found.

The Executive was designed to take advantage of the modularity of ARIAN II. The five functions of the Executive exist as small linear blocks of code within the Executive itself or as subroutines within ARIAN II. Additionally, the command routines themselves exist as subroutines, either memory-resident or on disk. Hence, the Executive is very small and consists mainly of subroutine calls.

The body of this chapter describes these five sections of the ARIAN II Executive is detail.

1 The Executive Reset

The purpose of the system reset in the Executive is to stabilize the environment of ARIAN II. ARIAN II does not use interrupts, so the setting of interrupt mode zero and the disabling of interrupts ensures that the ARIAN II Executive will not be interrupted by a maskable interrupt. ARIAN II contains a trap for non-maskable interrupts which returns control to the Executive.

ARIAN II typically uses two stacks -- one for the ARIAN II system and one for the user. In this way, the user can employ the program debugging facilities by executing a section of his code, returning control to ARIAN II and examining registers, etc., and then continuing the execution of his code with a minimum of difficulty. Hence, the Executive's function of resetting the ARIAN II system stack ensures that the user's stack will not be altered by the execution of ARIAN II routines.

Secondly, abnormal termination of an ARIAN II routine, such as in the case of an error, may result in an unbalanced system stack. The feature of the

Executive of resetting the system stack on each pass through the Executive ensures that the stack is balanced.

2 The ARIAN II Prompt

The prompt of ARIAN II serves two purposes: (1) it informs the user of the status of the disks and (2) it tells the user that ARIAN II is ready to receive a command.

The prompt consists of two digits followed by a greater-than sign ('>'). ARIAN II is designed to be used as a multiple-drive system, and two drives are of particular interest to ARIAN II. One is the Command Drive, and the other is the Logged-in Drive. The Command Drive holds a disk which contains the transient routines executed by ARIAN II. These are called the Level 3 commands, and this feature of transient routines is referred to as Command Level 3. If Command Level 3 is engaged and the ARIAN II Executive does not find the name of the current command in its local command directories, it will load the directory of the Command Drive and search for a Level 3 command of the specified name. If an entry is found, it will be loaded and executed. Level 3, as well as Level 1 and Level 2, commands will be discussed later in this chapter.

The two digits in the prompt give the numbers of the Command and Logged-in Drives, in that order. ARIAN II supports up to four drives, and the first digit, which gives the number of the Command Drive, may take on values from 0 to 4, while the second digit, which gives the number of the Logged-in Drive, may take on values from 1 to 4. If the number of the Command Drive is 0, then Command Level 3 is turned off; otherwise, the given number is the number of the drive from which Level 3 commands will be loaded.

Examples of the ARIAN II prompt are:

01> -- Command Level 3 is off, 1 is the number of the Logged-in Drive

22> -- 2 is the number of both the Command Drive and the Logged-in Drive

41> -- 4 is the Command Drive and 1 is the Logged-in Drive

3 The ARIAN II Input Line Editor

All typing done while the user is in ARIAN II with the exception of the immediate prompts is processed by the ARIAN II Input Line Editor. This editor collects each character as the user types it, stores it in a buffer, and allows the user to correct any typing errors he has made. When the user terminates the line by typing a carriage return, this buffer is terminated and ARIAN II processes the contents of the buffer. This routine is used while ARIAN II is in Command Mode and Block Line Entry Mode, and it may be called explicitly by the user (see the chapter on the ARIAN II assembler).

As each character is typed, it is checked to see if it is an editor control character. If it is, the function of the control character is executed; if not, the character is saved in the input line buffer and echoed to the principal I/O device. When the user has finished typing the line, he terminates it with a carriage return. No character is echoed as a result of this, and control is returned to the calling program.

The following is a list of all the editor control characters:

1. the Escape (<ESC>) key. When typed, <ESC> is printed on the principal I/O device as a dollar sign ('$') followed by a <CR>. This key tells the editor to delete the line typed so far and start over with a new line.

2. the Line Feed (<LF>) key. This key echoes as a <CR> and does not affect the line contained in the input line buffer. The sole purpose of this function is to allow the user to continue typing his line on the next physical line of the principal I/O device. No character is entered into the buffer as a result of this key, so the user must ensure that he does not run characters together.

3. the Tab (<TAB>) or Ctrl-I key. This key causes the cursor to tab to the next tab stop. As the cursor is tabbing, spaces are copied into the input line buffer. The tab stops are set by the TABS command, which is discussed in detail in Appendix I.

4. the Backspace (<BS>) or Ctrl-H key. This key allows the user the delete the last character he typed. It echoes as the cursor backing up to the previous position, and it is designed to be used with output devices which perform this function when they receive the ASCII code for a backspace. For example, if the user typed "ABCD<BS>", only "ABC" is in the input line buffer; the "D" has been deleted. If <BS> is typed again, the "C" is deleted, and so on. The user cannot delete beyond the beginning of the line; if he attempts to do this, an <ESC> is processed, echoing as a "$" <CR>.

5. the Delete (<DEL>) or Rubout key. This key performs the same function that Backspace does, but it echoes differently. The deleted characters are enclosed in backslashes. For instance, if the user typed "ABCD<DEL>E", this would be echoed onto the principal I/O device as "ABCD\D\E", indicating that the "D" was deleted and the string in the input line buffer is "ABCE". If the user types more than one <DEL> in a row, all the deleted characters are enclosed in one set of backslashes. For example, if the user types "ABCDE<DEL><DEL><DEL>ABE<DEL>C", this will appear on his terminal as "ABCDE\EDC\ABE\E\C", indicating that "EDC" and then "E" were deleted and the resulting string is "ABABC". This feature is provided to permit ease in the use of an I/O device that does not have a hardware backspace feature.

6. the Carriage Return (<CR>) key. The <CR> key always instructs the input line editor to terminate the input of the line and give the line to the calling program (ARIAN II) to interpret.

The input line editor is an extremely useful tool, and with practice it will soon become a very easy and natural tool to use.

4 The ARIAN II Command

The fourth section of the ARIAN II Executive is the command line parser. This subroutine breaks apart the command line into its various components and stores each element of the command in its appropriate internal buffers.

The ARIAN II Command is divided into five basic divisions:

1. the name of the command,
2. up to three "special" characters following the name of the command,
3. the name of a file or symbol,
4. one or two strings, and
5. up to three decimal or hexadecimal numeric arguments.

Only the first element is required, and any number of the rest of the elements are optional depending upon the nature of the command to be executed. These elements, however, must occur in the order specified; that is, the command name must be first, the special characters next, the file name next, etc. Fields 2, 3, 4, and 5 are separated by one or more delimiters, and if more one numeric argument is specified, these must also be separated by delimiters. If two strings are specified, they may optionally be separated by delimiters. The delimiter characters in ARIAN II and spaces and commas; these may be used as one prefers to improve readability.

The name of the command consists of from one to four alphanumeric characters, the first of which must be alphabetic. There is one special case in which the command is a line number, and, in this case, the other fields don't exist and the text which follows the line number is a space followed by the text of the line (see Appendix I).

The special characters are options pertaining to the individual commands. If special characters are used, the command name must consist of exactly four characters. These special characters then appear as the 5th, 6th, and 7th characters of a string. The ARIAN II commands look for these characters specifically, and if the special characters typed do not match those recognized by the command they are simply ignored.

The file name is a string of from one to eight alphanumeric characters, the first of which must be alphabetic. It is separated from the command name and its special characters by one or more delimiters.

The strings are vectors of characters enclosed in double quotes ("). If two strings are specified they need not be separated by any delimiters; the double quotes which close the first string and open the second serve this purpose. An escape character is provided by the ARIAN II Command Parser to permit the user to specify a double quote as part of a string; this is the backslash (\). The backslash instructs the parser to interpret the following character literally, so \" translates into one double quote and \\ translates into one backslash. If there are not more characters following the last string in the command line, it need not be terminated by a double quote; the <CR> which terminates the line is sufficient.

The numeric arguments consist of from one to five decimal and hexadecimal digits, the first of which must be decimal. The arguments are separated by one or more delimiters.

To facilitate ease of use in text processing applications, the alphabetic

characters in the command name, the special characters, the file name, and the numeric arguments are converted internally to upper case by the ARIAN II Command Parser. Therefore, the user may enter his commands in either upper- or lower-case, and they will be interpreted in upper-case. Of course, no translation is done on the contents of the strings. See Appendix I for a detailed description of the ARIAN II Command in SDL.

A command line beginning with an asterisk, semicolon, blank, or any ASCII character less in value than the character for zero is considered to be a comment and is not processed by ARIAN II. Upon encountering such a character, the Executive loops back upon itself and prepares to receive the next command line.

5 The ARIAN II Command Levels

After receipt of a command from the parser, ARIAN II then searches through up to three directories of commands for the specified command. These are called Command Levels, and are named Command Level 1, Command Level 2, and Command Level 3, in the order they are searched. The specified command is searched for at Command Level 1 first and Command Level 3 last.

Command Level 2 contains the ARIAN II Executive, or resident, commands. They are discussed in detail in Appendix I. These commands provide the basic control, file editing and manipulation, utility, and debug facilities of ARIAN II.

Command Level 1 is the Customized, or user-defined, command level of ARIAN II. These are created by the CUST Level 2 command, and it allows the user to specify the name of a command and its execution address in memory. This is a particularly useful feature, and the user can employ this to quickly and with a minimum of effort execute test programs or override any Level 2 command. The command itself can take the form of a simple subroutine, and is therefore very easy to create. Refer to Appendix I and the chapter on Level 1 and Level 3 commands for more information.

Command Level 3 is the disk-resident command level. Again taking the form of simple subroutines, these commands reside as files on disk and are loaded and executed by ARIAN II.

When ARIAN II receives a command, it first searches through the Level 1 directory. If it is found, it is executed and control is returned to ARIAN II. If it is not found, ARIAN II then searches through the Level 2 directory. Again, if it is found, it is executed and control is returned to ARIAN II.

If the command is not found at Level 2, ARIAN II checks to see if Level 3 is engaged. If not, an error message is given and control is returned to ARIAN II. If so, ARIAN II loads the directory of the Command Drive and searches it for the command file. If found, this file is loaded at its execution address, executed, and control is returned to ARIAN II. If it is not found, an appropriate error message is given and control is returned to ARIAN II.

Level 3 files are binary files. Their directory entries contain the name of the command with a ".CMD" extension, like "NAME.CMD", and an execution address. Refer to the chapter on Level 1 and Level 3 commands for more information.

CHAPTER 3

The ARIAN II FILE SYSTEMS

ARIAN II supports three types of files -- local text files, local binary files, and disk files (both text and binary). This chapter describes the techniques for creating and manipulating files in these three systems and the characteristics of files in these three systems.

Local files are memory-resident, while disk files reside on disk. ARIAN II maintains two local file directories which are used to assist the memory manager in its function of monitoring and controlling the use of memory in the ARIAN II environment and the user in identifying his resident files. Each disk also contains a directory of its files, and this directory is loaded into memory and analyzed whenever a file is transferred to or from disk.

The ARIAN II user may create up to ten local text files and ten local binary files. Each disk is capable of holding up to 64 disk files. ARIAN II partitions memory for various functions, and the ARIAN II workspace is the partition of memory within which the local text files reside. Binary files may exist anywhere within the address space, as may Level 3 command files. Most Level 3 command files, however, load and execute in the transient program area (refer to the chapter on Level 1 and Level 3 commands).

1 Local Text Files

The local text files are structured organizations of the user's text which reside in memory. They consist of lines of text and are terminated by an end-of-file mark. Each line consists of a four-digit line number, a space, and the text of the line. Transparent to the user, each line also contains some diagnostic information which helps ARIAN II determine as to the validity of the file.

One of the local text files is designated as the primary file, while the others are referred to as secondary files. The primary file is the file which is currently being referenced by the user. The file and line editing commands, as well as the disk transfer commands, operate on the primary file. The secondary files reside in memory for the purpose of later being included in the primary file or being designated as the primary file.

When a local file is created, it is automatically made primary. ARIAN II can create local text files in a number of ways, but the most common way to initially create a local text file is by using the FILE command. "FILE FILENAME" will create a local text file starting at a memory location selected by the memory manager and make it primary. Once created, the common way of entering information into this file is by using the "APND" command. APND

appends the following block of lines, entered in Block Line Entry Mode, to the end of the current primary file, or, if a line number is specified, after the given line. The Block Line Entry Mode is terminated by the escape sequence consisting of a Ctrl-C followed by a <CR>. Upon exiting this mode, ARIAN II automatically renumbers the file starting at 5 and incrementing by 5 unless the N option is used ("APNDN"), in which case each line in the appended block begins with the specified line number or 9999 if the no line number was given.

The primary file editing commands, namely <lnum>, APND, DEL, EDIT, FIND, ISRT, and RNUM, make up the resident file editor of ARIAN II. ISRT, or insert, is the same as APND, but it inserts the block of lines before the specified line. DEL is used to delete one line or a block of lines, <lnum> is used to enter a line directly by typing a 1-4 digit line number followed by a space and the text of the line, EDIT is the intra-line editor, FIND is used to search for a specified string in the primary file, and RNUM renumbers the primary file.

All of these commands make the resident primary file editing facility of ARIAN II very responsive to the user. He may intermix these commands with the other system commands at his discretion.

For example, the following illustrates the use of some of these primary file editing commands. This is not a precise example; all ARIAN responses are in lower case and user entries are in upper case. Refer to the sample session for more detailed and precise examples.

```
01>FILE TEST
test     2a00  2a00
01>APND
?THE RAIN IN SPAIN
?FALLS MAINLY
?ON SUNDAY THROUGH MONDAY
?ON ODD WEEKDAYS
?IN THE PLAIN.
?^C
01>LIST
0005 the rain in spain
0010 falls mainly
0015 on sunday through monday
0020 on odd weekdays
0025 in the plain.
01>DEL 15,20
01>LIST
0005 the rain in spain
0010 falls mainly
0025 in the plain.
01>RNUM
01>LIST
0005 the rain in spain
0010 falls mainly
0015 in the plain.
01>FIND "SP
0005 the rain in spain
01>LIST 15
0015 in the plain.
01>RNUM 10,10
01>LIST
0010 the rain in spain
```

```
0020 falls mainly
0030 in the plain.
01>25 WHEN IT.WISHES
01>LIST
0010 the rain in spain
0020 falls mainly
0025 when it wishes
0030 in the plain.
01>
```

As the user can see, the resident primary file editor of ARIAN II is quite versatile. There are also a number of Level 3 commands which supplement this editor, but they will be discussed later. Also, not every aspect of these commands has been discussed. Each command has several forms with several options, and Appendix I discusses them at some length.

The EDIT command invokes an intra-line editor which can be used to change the contents of a line of the primary file without retyping the entire line. It contains subcommands which permit the user to delete, replace, and insert characters into the line. EDIT is discussed in much greater detail in Appendix I.

2 Local Binary Files

The local binary files are unstructured organizations of programs and data which reside in memory. They are defined solely by their entries in the local binary file directory, and this entry only provides information as to the name of the file and the range of memory over which it exists. The memory managers of ARIAN II totally ignore the entries in the binary file directory, and these files are subject to destruction by the memory managers.

For this reason, it is recommended that the user create all binary files outside the local text file workspace and transient program areas of ARIAN II. Also, by realizing that the local text files in the ARIAN II workspace grow upward in memory, the user may judiciously create his binary files in the upper regions of the workspace. This, of course, must be done with care and consideration of his future actions in the system.

Local binary files are created in one of three ways: (1) implicitly when ASSM is used with a file name specification, (2) implicitly when a binary file is loaded from disk, and (3) explicitly by the user via the FILEB option of the FILE command. When created by the assembler, the binary file is defined to exist in the space of memory covered by the assembly from the last ORG or the entire assembly range if no ORG is specified within the assembled file. The LDIRB command lists all the binary files, and the first line of this directory listing gives this range as two hexadecimal numbers. When created by the LOAD command, the binary file has a range of an integral number of blocks. It starts at the load address of the file and extends over the size of the file as it resides on disk. Finally, when created explicitly, the binary file has exactly the range specified by the user.

No management is done of the binary files by ARIAN II. These files are referenced solely for the convenience of the user, and it is up to him to use them at his discretion.

3 Local File Manipulation

A number of Executive commands are available through ARIAN II which manipulate, either directly or indirectly, the local files and provide information to the user on their validity.

Two commands, FCHK and RCVR, apply only to local text files. FCHK is used to determine the validity of a local text file. Employing the inherent structure of the ARIAN II text file, FCHK insures that the specific data of each line of the specified or implied file is valid. FCHK with no argument validates the primary file, and FCHK FILENAME validates the specified local text file without affecting the status of the other files. The other command, RCVR, is used to attempt to recover a local text file that has been deleted from the local text file directory. It uses the inherent structure of the ARIAN II text files to attempt to determine the bounds and other pertinent information on the data starting at the address specified by the command in an attempt to rebuild the directory entry. If the file is found to be intact up to its end-of-file mark, a directory entry is created. The format of this command is RCVR FILENAME ADDRESS, where FILENAME is the new name of the file to be recovered and ADDRESS is the starting address from which to attempt the recovery.

The rest of the commands are similar in structural format. They are FILE, LDEL, LDIR, LNAM, and LSCR. Each command as given operates on the local text files, while FILEB, LDELB, LDIRB, LNAMB, and LSCRB operate on the local binary files.

The FILE command is used to create local files, and, in the case of text files, it can also specify the bounds and name of the current primary file. LDEL deletes the specified local file, and LSCR scratches (deletes all the files in) the specified local directory. In the cases of LDEL and LSCR, only the directory entries are affected; note that RCVR may be used to restore the deleted text files. LDIR lists the contents of the specified local directory, and, finally, LNAM is used to rename the specified local file.

4 Disk Files

Disk files are files which reside on disk; they may be either text (type 0) or binary (type 1). All disks contain directories which define the files contained on them, and these directories provide the only direct means of definition of such files. The directory entries contain the name of the file, its size in 256-byte blocks, its type, its starting disk address, and, if it is a binary file, its execution/load address.

The files themselves are organized collections of data stored in sequential 256-byte blocks on disk. They contain no particular distinguishing features, except for text files, whose internal organization is exactly the same as local text files. All disk files are loaded and stored sequentially to and from memory.

For this purpose, ARIAN II supports the LOAD and SAVE commands. LOAD will load the specified disk file into memory and make the appropriate entry in either the local text or binary file directory. SAVE will save the current primary file on disk under the name specified, and SAVEB will create and store a binary file on disk.

In addition to the LOAD and SAVE commands, ARIAN II supports three Executive commands which perform functions similar to LDEL, LDIR, and LNAM, but

they operate on disk files. These commands are DDEL, DDIR, and DNAM. DDEL deletes the specified disk file's directory entry, DDIR displays the disk file directory, and DNAM renames the specified disk file.

As with LOAD and SAVE, DDEL, DDIR, and DNAM operate on the Logged-in drive. As mentioned earlier, the number of the Logged-in drive is specified as the second digit of the ARIAN II prompt. As with all ARIAN II Executive commands, refer to Appendix I for a detailed description of their function.

The following are examples of the use of some of the ARIAN II commands discussed in this chapter. Again, refer to Appendix I for more information on these commands. The upper/lower case conventions of user/ARIAN II apply as before.

```
01>DDIR
fdos      10 1 2000        sdl       44 0
arain2    60 0             arian3    50 0
01>LDIR
test     2a00  2a6b  0050
01>LNAME TEST
new name? TEST1
01>FILE
test1    2a00  2a6b  0050
01>LDIRB
b800 b85a
01>SAVE IT
$ file saved
01>DDIR
fdos      10 1 2000        sdl       44 0
arian2    60 0             arian3    50 0
it         1 0
01>DDEL IT
01>DDIR
fdos      10 1 2000        sdl       44 0
arian2    60 0             arian3    50 0
01>LSCR
01>LDIR
01>LDIRB
b800 b85a
01>
```

CHAPTER 4

LEVEL 1 and LEVEL 3 COMMANDS

Command Levels 1 and 3 are quite similar in many respects. The commands in these levels take the form of subroutines, i.e., they are usually bodies of code ending in a RET instruction. They are memory-resident during execution. Finally, they may take advantage of the various entry points into ARIAN II and use the assembler-defined symbols.

1 Command Level 1

Command Level 1 is the customized command level. These commands, which are created by the CUST Executive command, are memory-resident and are defined only by their entries in the Customized Command Table. These table entries consist of the name of the command (1 to 4 characters, the first of which must be alphabetic) and the execution address of the command.

Customized commands are created by the CUST <cname> <hadr>? variant of the CUSTomize command. The subroutine starting at the specified or implied address becomes the new command, and, until the user deletes this customized command, whenever he types the command name given, he will execute this subroutine. If no address is specified, the default execution address is used to define the starting address of the command.

Examples of the use of the CUST command are:

CUST LIST
The LIST Executive Command of ARIAN II is redefined, and the user will execute the subroutine starting at the default address whenever he types "LIST".

CUSTD LIST
The user-defined LIST command is deleted, and the normal system LIST command is restored.

CUST TEST 6C00
A new customized command called TEST is created; its

execution starts at location 6C00 hexadecimal.

CUSTL
All the customized commands defined by the user that are currently in effect are listed with their execution addresses.

CUSTS
All customized commands are scratched (deleted).

2 Command Level 3

The Level 3, or disk-resident, commands take the same form as the Level 1 commands, but they are loaded from disk when their command name is given. They reside on disk as binary (type 1) files whose names consist of the four characters of the command name followed by ".CMD", like "HELP.CMD". The load address in their disk directory entries specifies the memory location at which they are loaded and executed.

Level 3 commands may execute anywhere in memory, but ARIAN II has reserved an area of memory for these commands. It is called the transient program area, and it is the 1K section of memory which starts at 1K above the end of the ARIAN II workspace; it usually starts at BC00 hexadecimal. Most Level 3 system commands execute in the transient program area, but there are exceptions. It is recommended that the user assemble all his Level 3 commands to execute in this region.

Like Level 1 commands, Level 3 commands can be created by creating a file of the program and assembling it (by using ASSM 0BC00 -- see the chapter on the assembler). The assembler will give the assembly limits specified by the last ORG and the end of the program, and, if these are the actual limits of the program, the user can save it on disk as a Level 3 command by issuing the SAVEB command, like "SAVEB NAME.CMD 0BC00 0BDEF". This will save the binary of the command on the logged in drive. To execute the command, the user should ensure that this drive is made the command drive and then type the name of the command, "NAME". It will then be loaded at its load address (BC00 hexadecimal in this case) and executed.

3 Interfacing Level 1 and Level 3 Commands to ARIAN II

In order to interface Level 1 and Level 3 commands to ARIAN II, the user must understand how ARIAN II parses commands. Commands are parsed into several distinct subfields, as mentioned earlier, and the elements of these subfields are stored in specified buffers within the ARIAN II scratchpad RAM area. All commands start with a command name followed by up to three special characters and any number of arbitrary characters. The command name is always interpreted as being four characters long, with unspecified characters being spaces. It is converted to upper-case letters and placed in the four-byte buffer CBUF by the parser. The special characters are placed in the three one-byte buffers, SPCHR1, SPCHR2, and SPCHR3, which immediately follow CBUF.

The second command subfield is a file or command name. This field consists of up to eight characters, the first of which must be alphabetic. It, too, is

internally capitalized by the parser. This field is stored in FBUF, left justified and blank-filled on the right. If no element appears in this field, FBUF will contain only binary zeroes (0 hexadecimal).

The third command subfield consists of the two strings. Each string may consist of up to 40 characters, and they are stored in SBUF1 and SBUF2. No upper-case conversion is done to string fields. If a string is stored in one of the string buffers, the first character of the buffer will be a double quote and the string is terminated by a carriage return character (0D hexadecimal). The ARIAN II parser permits the first string to be over 40 characters, in which case it will run into the second string buffer. In this case, there must be no second string, or the first string will be truncated.

The fourth command subfield is the numeric argument field. This field consists of from one to three numbers. The arguments parsed in this field are placed in ABUF (four ASCII characters per argument) and BBUF (two bytes per value). ABUF contains the ASCII characters of the numbers in groups of four (i.e., ABUF to ABUF+3 contains the first number, ABUF+4 to ABUF+7 contains the second, and ABUF+8 to ABUF+11 contains the third); if the numbers contain more than four digits, only the first four are contained in these fields, and if they contain less than four digits, these fields are left-filled (normalized) with the character for zero. BBUF contains their values, assuming they are hexadecimal numbers, in groups of two (i.e., BBUF and BBUF+1 contain the first value in the INTEL-standard low-order/high-order format, BBUF+2 and BBUF+3 contain the second, and BBUF+4 and BBUF+5 contain the third). If there are fewer than three numeric arguments specified, the corresponding ABUF and BBUF fields will contains binary zeroes (0 hexadecimal).

Hence, with the arguments of a customized command parsed in this manner, the user can create customized commands which perform like the normal ARIAN II Executive Commands. ABUF is usually used to contain line numbers for reference, BBUF is usually used to contain values used to specify hexadecimal quantities, SBUF1 and SBUF2 contain the strings, FBUF is usually used to contain a file or command name, SPCHR2, SPCHR2, and SPCHR3 contain the special characters, and CBUF contains the command name.

The entire input line, as a consequence of the input line editor, is available to the user in IBUF. The first character of the line is at IBUF, and the line ends in a <CR>. Location IBUF-1 contains a count of the number of characters in the line, including the character at IBUF-1 and the ending <CR>.

The following are examples of the use of these buffers. Note that <null> refers to the binary zero (0 hexadecimal), and <null>s refer to the number of binary zeroes required to fill the specified field.

```
     REGS BC 2000
     CBUF = "REGS", SPCHR1=SPCHR2=SPCHR3=<null>, FBUF =
"BC" and 6 blanks, SBUF1=SBUF2=<null>s,
ABUF(0-3)="2000", ABUF(4-11)=<null>s, BBUF(0)=0,
BBUF(1)=20 Hexadecimal, BBUF(2-5)=<null>s

     xdirx test
     CBUF = "XDIR", SPCHR1="X", SPCHR2=SPCHR3=<null>,
FBUF = "TEST" and 4 blanks, and the rest of the buffers
contain <null>s.

     saveb1 test.cmd 0a000 0afe0
     CBUF = "SAVE", SPCHR1="B", SPCHR2="1",
SPCHR3=<null>, FBUF = "TEST.CMD", SBUF1=SBUF2=<null>s,
```

ABUF(0-3) = "0A00", ABUF(4-7) = "0AFE",
ABUF(8-11)=<null>s, BBUF(0) = 0 hex, BBUF(1) = A0 hex,
BBUF(2) = E0 hex, BBUF(3)= AF hex, and
BBUF(4-5)=<null>s

The addresses of CBUF, ABUF, BBUF, FBUF, SPCHR1, SPCHR2, SPCHR3, SBUF1, and SBUF2 are given in Appendix II.

4 ARIAN II Entry Points

Aside from using a simple RET instruction to return to ARIAN II from a Level 1 or Level 3 command, ARIAN II has three reentry locations which may be branched to for a clean return to the system.

These reentry points are at memory locations 0, 4, and 66 hexadecimal. Location 0 is the ARIAN II dead start entry point; the entire ARIAN II system is initialized if execution begins at this point. Location 4 is the ARIAN II warm start entry point; only part of the system is initialized if execution begins at this point. Finally, location 66 hex is the non-maskable interrupt entry point. It transfers control directly to the ARIAN II Executive; no initialization is done if the user enters here. Location 66H is also referenced by the assembler-defined symbol ZEOR.

5 Use of Restart Locations by ARIAN II

Restarts 2, 3, 4, 5, and 6 are unused by ARIAN II and are free for the user to employ at his discretion.

Restart 0 is used for the dead and warm restarts of ARIAN II, and it is recommended that it not be used for any other purpose. Restart 1 is the ARIAN II breakpoint reentry restart, and it may be employed by the user if he does not wish to set any breakpoints under the BREK command. Finally, Restart 7 is the ARIAN II memory trap, and it, too, may be employed by the user at his discretion. This memory trap is a safety feature to halt programs which attempt to run in non-existant memory.

6 The ARIAN II Jump Table

Appendix II also summarizes the functions supported by the jump table in ARIAN II. Briefly, this jump table provides easy access to the following routines and Executive commands:

1. CKA11 - check for the presence of at least one numeric argument. Return with zero set if no numeric arguments, clear if at least one.
2. CKA21 - check for the presence of at least two numeric arguments. Return with zero set if two arguments are not present, clear if at least two are present.
3. CKFN1 - check for the presence of an argument in FBUF. Again, zero set means no argument, clear means there is an argument.
4. DSCAN - scan logged in disk directory for the

file name contained in FBUF. If found, return with zero set and HL pointing to the first byte of the directory entry; otherwise, return with zero clear.

5. FILE - execute the FILE Executive command. Arguments are passed in the appropriate buffers.

6. FIND - find the line whose number is contained in ABUF (normalized). HL point to the first byte of the line upon return. Line number found is first line greater than or equal to the number in ABUF.

7. LINE - execute the <lnum> Executive command. The line contained in IBUF is entered into the primary file.

8. LOAD - execute the LOAD executive command. Arguments are passed in the appropriate buffers.

9. SAVE - execute the SAVE executive command. Arguments are passed in the appropriate buffers.

10. MESS - print the character string pointed to by HL and ending in <CR> on the principal I/O device and return to the ARIAN II Executive. This routine is good for printing fatal error messages.

11. FSEA - search the local text file directory for the file whose name is in FBUF. A number of flags are set by this entry; see Appendix II. Zero flag is clear if file name is found, set if not found.

12. RNUM - execute the RNUM executive command. Arguments are passed in the appropriate buffers.

CHAPTER 5

The ARIAN II ASSEMBLER

The assembler of ARIAN II is a very powerful real-time assembler based on the INTEL-standard mnemonics for the 8080 microprocessor. The assembler, however, is not just an 8080 assembler; it is a pseudo-Z80 assembler which gives the user the ability to assemble some of the common Z80 instructions as well as all of the 8080 instructions.

The assembler features its own set of pseudo-ops (most of which are similar to the INTEL-standard pseudo-ops), all the 8080 mnemonics, some arithmetic operations in the operand field, literal character definitions in the operand field, a unique set of Z80 instructions, the ability to explicitly create binary files, and manipulation of the default execution address.

The standard features supported by the assembler include:

1. free-format source input,
2. symbolic addressing, including forward and relative symbolic references,
3. up to 384 six-character symbols,
4. reserved names for the registers which may be redefined,
5. a full set of pseudo-ops, including default execution address control, and
6. automatic generation of a symbol table which may be referenced later.

1 The Assembler in General

The assembler translates the lines contained in the primary file into object code residing in memory. The second character following the line number is the first source code character position. Therefore, the character immediately following the line number should be a space; the APND and ISRT commands place a space here automatically, and the user need only be concerned with this restriction if he enters his own lines using the <lnum> <text> command. Line numbers are not processed by the assembler; they are merely reproduced in the listing.

The assembler will assemble a source program file composed of statements, comments, and pseudo operations on each line. It does this in two passes. During Pass 1, the assembler allocates all storage necessary for the translated program and defines the values of all symbols used by creating a symbol table.

The storage allocated for the object code will begin at the byte explicitly or implicitly specified by the ASSM command unless an ORG pseudo-op is present in the program. During Pass 2, all expressions, symbols, and ASCII constants are evaluated and placed in allocated memory in the appropriate locations. The listing, also produced during Pass 2, indicates exactly what data is in each location of memory.

Statements contain either symbolic ARIAN II assembly language instructions or pseudo-ops. These statements are divided into up to four fields: (1) a name field (optional), (2) an operation field, (3) an operand field (optional), and (4) a comment field (optional). If the first valid character in a statement is a semicolon or asterisk, the entire statement is processed as a comment.

The name field, if present, must begin in the first assembler character position in the input line; this is the second character after the line number. The symbol in the name field can contain as many characters as the user wishes, but only the first six characters are used in the symbol table to uniquely define the symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters. Digits are allowed. The name field may or may not be terminated by a colon, at the user's discretion.

The operation field contains either an ARIAN II assembler operation mnemonic or a system pseudo-op. The ARIAN II assembler operation mnemonics and system pseudo-ops are described below. The operation field is separated from the name field by one or more blanks; if no name field is present, it begins in or after the third character position after the line number (at least two spaces in front of it).

The operand field contains parameters pertaining to the operation in the operation field. If two arguments are present, they must be separated by a comma. The operand field is separated from the operation field by at least one space.

The comment field is for explanatory remarks. It is reproduced in the listing without processing. Comment lines must start with either a semicolon or an asterisk; it is recommended that comments at the end of a statement also start with one of these characters, but this is not a restriction (comments after the operand or operation field may or may not start with a semicolon or asterisk).

The ARIAN II assembler is completely free-format, as described above, but a "standard" ARIAN II assembler program line format is defined by ARIAN II and is recognized by various options of several commands. This format is as follows:

1. the name field must start in the first assembler line column,
2. if the name field exists, the operation field starts one space after the colon or end of the name field; if the name field does not exist, the operation field starts in the second assembler line column,
3. the operand field starts one space after the operation field,
4. the comment field starts one space after the operand field if there is one or one space after the operation field if there is no operand field, and the comment field must start with a semicolon, and
5. if the entire line is a comment, it must start with either an asterisk or a semicolon (asterisk preferred) in the first assembler line column.

Symbolic names and addressing are also supported by the assembler. To assign a symbolic name to a statement, the name is placed in the name field. To leave off the name field, the user skips two or more spaces after the line number (one or more spaces in block line entry mode) and begins the operation field. If a name is attached to a statement, the assembler assigns it the value of the current location (program) counter. The program counter holds the address of the next byte to be assembled if the instruction is a machine instruction or pseudo-op. The EQU pseudo-op, however, assigns to its label a value which is defined in the operand field. Note: do not confuse the location counter of the assembler with the "$" symbol discussed later; this location counter points to the next instruction to be assembled, while "$" points to the instruction after the current instruction if the current instruction is a normal mnemonic or "$" points to the current instruction if it is a pseudo-op.

Names are defined when they appear in the name, or label, field. All defined names may be used as symbolic arguments in the operand field. The reserved system symbols, however, are defined by the assembler and must not be redefined by the user; a duplicate label error will result if this is done. These reserved system symbols are discussed later.

In addition to the user-defined and system-defined symbols, the assembler has reserved several symbols to represent the registers of the 8080. These symbols, like the system reserved symbols, may only be used in the operand field. These symbols are:

1. A -- the accumulator; value 7,
2. B, C, D, E, H, and L -- the B, C, D, E, H, and L registers; values 0, 1, 2, 3, 4, and 5, resp.,
3. M -- memory (pointed to by H&L); value 6,
4. P, PSW -- the program status word; value 6, and
5. S, SP -- the stack pointer; value 6.

The assembler also supports relative symbolic addressing. If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset. All defined symbols, including "$", may be used in this relative symbolic addressing scheme.

For example, LDA $+5 loads the accumulator with the value of the byte located five bytes after the beginning of the next instruction. Also, SSPD LOC-7 stores the value of the stack pointer starting at the byte located seven bytes in front of the memory location pointed to by the symbol "LOC".

The assembler permits the user to write positive and negative numbers directly in a statement. They will be regarded as integer constants, and their binary values will be used appropriately. All unsigned numbers are considered to be positive. Decimal constants can be defined using the suffix "D" after the numeric value, but this is not required since the default is decimal. Hence, 10 and 10D define the constant ten decimal. Hexadecimal constants must start with a digit and end with the suffix "H". Examples of hexadecimal constants are 10H, 0AFH, 000101H, and 00BCH.

ASCII constants may be defined by enclosing the ASCII character within single quotes, i.e., 'C'. Two characters may be enclosed within single quotes for double-word constants.

2 Assembler Pseudo-ops

The following is a list and a description of the pseudo-ops recognized by the assembler:

1. ASC '<string>' -- ASCII string. This pseudo-op loads consecutive memory locations starting at the current value of the location counter with the ASCII values of the characters specified in the string.
2. DB <expression> -- define one byte. This instruction evaluates the specified operand and loads one 8-bit value into the location pointed to by the location counter. If the number is evaluated into a 16-bit quantity, only the low-order byte is loaded.
3. DS <expression> -- define storage. This reserves the specified number of bytes starting at the current value of the location counter.
4. DW <expression> -- define one word. This instruction evaluates the specified operand, producing a 16-bit value which it loads into memory (low order, high order) at the location pointed to by the location counter and the succeeding location.
5. END -- end the assembly. This statement is not required; assembly will stop when the end of the file is reached or this statement is encountered.
6. <label> EQU <expression> -- the specified label is assigned the computed value of the operand; the computed value is a 16-bit quantity.
7. EXEC <expression> -- the default execution address is set to the value of the expression.
8. LST -- turn on the listing of the assembly of the program on the principal I/O device if it is not already on. This can be used to list only selected portions of a program during the assembly.
9. NLST -- turn off the listing of the assembly of the program.
10. ORG <expression> -- set the origin (location counter) to the specified value. This instruction also resets the assembly limits and the location in memory at which the object code is loaded. If an ORG appears more than one time in the program, the limits set by the last ORG and the end of the assembly are reflected in the assembly limits displayed by the LDIRB command.

All pseudo-ops may be preceeded by a label.

3 System Reserved Labels

Another feature of the ARIAN II assembler is its system reserved labels. These symbols provide easy access to a host of utility subroutines for functions such as I/O, data conversion, and ARIAN entry points, and they also supply some commonly-used buffer and variable addresses. All system reserved symbols start with the letter "Z" and are at most four characters long.

The following is a complete list of the ARIAN II assembler reserved symbols. They are described in detail in Appendix III.

Symbol	Function
ZEOR	the Executive reentry point to ARIAN II
ZLIN	the ARIAN II input line editor
ZINK	polls the principal I/O channel for an <ESC>
ZIN	input one character from principal I/O channel
ZOUT	output one character to principal I/O channel
ZCR	output <CR> <LF> to principal I/O channel
ZARG	the ARIAN II Executive Parser
ZHOT	display the A register as two hexadecimal digits
ZDOT	display the A register as up to three decimal digits, left blank fill
ZBLK	output space to principal I/O channel
ZPRH	print string pointed to by HL ending in <CR> on principal I/O channel
ZPPH	print string pointed to by HL ending in <CR> on printer
ZPRR	print string pointed to by return address ending in <null> (0)
ZPHL	print HL as four hexadecimal digits
ZSHD	compute BC = HL - DE
ZBOF	address of two-byte buffer which contains starting location of primary file
ZEOF	address of two-byte buffer which contains ending location of primary file
ZBBF	address of BBUF
ZIBF	address of IBUF
ZEN	exchange nybbles of the A register
ZCHA	convert low nybble of A to its ASCII hexadecimal equivalent
ZCAH	convert ASCII hexadecimal character in A to its binary equivalent in A
ZCRL	call relative long
ZJRL	jump relative long

4 The Standard 8080 Mnemonics

The ARIAN II assembler recognizes all the standard 8080 mnemonics. For reference, they are:

Mnemonic	Mnemonic	Mnemonic	Mnemonic
ACI	ADC	ADD	ADI
ANA	ANI	CALL	CC
CM	CMA	CMP	CNC
CNZ	CP	CPE	CPI
CPO	CZ	DAA	DAD
DCR	DCX	DI	EI
HLT	IN	INR	INX
JC	JM	JMP	JNC
JNZ	JP	JPE	JPO
JZ	LDA	LDAX	LHLD
LXI	MVI	MOV	NOP
ORA	ORI	OUT	PCHL
POP	PUSH	RAL	RAR
RC	RET	RLC	RM
RNC	RNZ	RP	RPE
RPO	RRC	RST	RZ
SBB	SBI	SHLD	SPHL
STA	STAX	STC	SUB
SUI	XCHG	XRA	XRI
XTHL			

5 The Special Z80 Mnemonics

The following is a list of the special Z80 mnemonics recognized by the ARIAN II assembler and their ZILOG equivalents.

Mnemonic & Operand	ZILOG Equiv	Comments
SSPD <expression>	LD (nn),SP	store SP direct
LSPD <expression>	LD SP,(nn)	load SP direct
SBCD <expression>	LD (nn),BC	store BC direct
LBCD <expression>	LD BC,(nn)	load BC direct
SDED <expression>	LD (nn),DE	store DE direct
LDED <expression>	LD DE,(nn)	load DE direct
EXA	EX AF,AF'	
EXX	EXX	
BR <expression>	JR n	branch relative
BC <expression>	JR C,n	branch relative on carry
BNC <expression>	JR NC,n	branch relative on no carry
BZ <expression>	JR Z,n	branch relative on zero
BNZ <expression>	JR NZ,n	branch relative on no zero
DBJ <expression>	DJNZ n	decrement B and branch relative
LD	LDD	
LDR	LDDR	
LI	LDI	
LIR	LDIR	
CD	CPD	
CDR	CPDR	
CI	CPI	
CIR	CPIR	
NEG	NEG	
RLD	RLD	
RRD	RRD	
SHB	SBC HL,BC	
SHD	SBC HL,DE	
SHS	SBC HL,SP	
AHB	ADC HL,BC	
AHD	ADC HL,DE	
AHS	ADC HL,SP	
IM0	IM 0	
IM1	IM 1	
IM2	IM 2	

```
ID                  IND
IDR                 INDR
II                  INI
IIR                 INIR

OD                  OUTD
ODR                 OTDR
OI                  OUTI
OIR                 OTIR

CIN                 IN A,(C)
COT                 OUT (C),A
```

6 Operand Evaluation

Operand evaluation in ARIAN II is performed from left to right, and there is no operator hierarchy. Parenthesized expressions are not permitted. All operations performed are done on sixteen-bit quantities, and all operators have both an infix and prefix form, the prefix form being the same as the infix form with the first operand having a value of zero. Single character strings of the form '<char>' are permitted in expressions and stand-alone.

The operators recognized by the ARIAN II Assembler are:

+ -- addition
- -- subtraction
& -- logical AND
! -- logical OR
% -- logical Exclusive OR
< -- Extract High (add operands, exchange bytes, and zero out high-order byte)
> -- Extract Low (add operands and zero out high-order byte)
* -- multiplication
/ -- division

All numeric arguments are assumed to be decimal unless the suffix "H" is appended to them. Therefore, 100 is 100 decimal and 100H is 100 hexadecimal.

The "$" symbol is used as the value of the location counter for the next instruction. In normal instrucions, "$" points to the first byte of the next instruction; in pseudo-ops, "$" points to the first byte of the pseudo-op. This permits relative addressing to take the form of "BR LABEL-$" and pseudo-ops like "STACK EQU $" to be used.

Finally, if an expression with a value greater than 0FF hexadecimal is loaded into an eight-bit register, like "MVI A,1FFH", only the low-order byte of this value is loaded.

Examples of permitted expressions include:

```
NCHRS*8
<START
SIZE/256+34-12&0FFH!20H>0*2
LABEL+3
POINT-'A'+60
```

```
POINT3-0AFH+6-2
HERE-$-2
```

7 Assembler Error Messages

The following is a list of the error messages produced by the assembler and their meanings:

Error	Meaning
R	register error. The register name is missing or invalid.
S	syntax error. The instruction syntax is incorrect.
U	undefined symbol. The referenced symbol is incorrect.
V	value error. The computed value cannot be represented as a 16-bit value or the instruction has a syntax error.
M	missing label error. A required label is missing.
A	argument error. The instruction's argument is of the wrong type or generally incorrect.
L	label error. The label of this instruction contains an invalid character.
D	duplicate label error. The label of this instruction has been defined elsewhere.
O	opcode error. The opcode (in the operation field) of this instruction is invalid.

CHAPTER 6

The ARIAN II SUBSYSTEMS -- TERMINAL and UTILITY

ARIAN II supports two subsystems of commands at the Executive level; they are the Terminal subsystem, invoked by the TERM command, and the Utility subsystem, invoked by the UTIL command. Each is a complete subsystem; they contain their own argument parsing schemes and set of commands.

1 The Terminal Subsystem

The TERMinal command invokes the Terminal, or inter-system communication, subsystem of ARIAN II. Under this subsystem the microcomputer becomes relatively transparent to the user, and the user's terminal is made to respond like a timesharing terminal to an external computer system. Each character typed is sent to a modem and acoustic coupler, and each character received by the modem is sent to the user's CRT.

The Terminal subsystem responds to three commands. They are Ctrl-A, Ctrl-L, and Ctrl-R. Ctrl-A invokes the terminal alternate command set. This command set consists of the following commands:

1. M -- transfer to the Monitor Command System (MCS).
2. R <hadr> -- call the subroutine located at the specified address. The return in the subroutine transfers control to the terminal mode (not the terminal alternate command set).
3. T <hadr> -- transfer the downloaded code to the specified address. This command transmits a <CR> to the external computer, and the object code downloaded is loaded into memory starting at the specified address. The addresses given in the code are ignored, and the entire code is loaded sequentially into memory.
4. F -- turn off echo; full duplex mode of operation
5. H -- turn on echo; half duplex mode of operation (default)
6. X -- exit to terminal mode.

All commands in the terminal alternate command set are parsed like MCS (Monitor Command System) commands. The parser scans the input line until it

encounters a non-blank character; this character is interpreted as the command name. The parser then scans until it encounters another non-blank character; it then interprets the string starting at this character as a hexadecimal number and scans until it encounters an invalid hexadecimal character. Only the last four characters are used to interpret the final value. If fewer than four characters are in the number then the number is zero filled from the left.

Ctrl-L is the download command to the terminal mode subsystem. A <CR> is transmitted to the external computer, and the object code, in INTEL-standard format, is loaded into the microcomputer's memory at the addresses specified in the load blocks. Downloading can be interrupted at any time by typing <ESC> on the principal I/O device keyboard. This key transfers control to the terminal mode subsystem.

Ctrl-R simply returns control to the program which called the terminal subsystem. This calling program is usually ARIAN II.

More information is available on the TERM command in Appendix I.

2 The Utility Subsystem

The UTIL command invokes the Utility subsystem. This subsystem gives the user the ability to examine and modify memory directly. In response to this command, the user is prompted with a slash. He may then enter any of the following UTILity commands:

1. C <hadr> <hadr> <hadr> -- copy the block of memory defined by the first two addresses to the location in memory starting at the third address. The original block is unchanged unless the third address resides within this block.
2. D <hadr>? <hval>* -- deposit the specified values into memory.
3. E (<hadr> <hadr>?)? -- examine a specific memory location or block of memory.
4. F <hadr> <hadr> <hval>* -- find all occurrances of the specified byte string in the specified memory block.
5. S <hadr>? -- set/display the default pointer.
6. X -- return to ARIAN Command mode.

These commands are also interpreted like MCS commands. If more than one value is specified in the command, successive values are separated by one or more spaces. <hadr> is a 16-bit quantity, and <hval> is an 8-bit quantity; <hadr> is an address, and <hval> is a value. Refer to Appendix I for more detailed information on the Utility subsystem. The Utility subsystem commands are derived from MCS V1.6; the most complete description of these commands is available in the "Monitor Command System User's Manual".

CHAPTER 7

SUMMARY of the ARIAN II COMMANDS

ARIAN II supports over 30 Executive, or Level 2, commands. Several of these commands have been mentioned so far, and Appendix I covers all of these commands in detail.

The Executive commands are grouped into nine catagories. They are:

1. System Control
2. Primary File Editing
3. Local File Control
4. Disk File Control
5. Local/Disk File Transfer
6. List/Print
7. Program Debugging
8. Assembler
9. Utility

1 System Control

The System Control commands are CMND, CONT, CUST, EXEC, EXIT, RESET, SETC, TABS, and TERM.

The CMND command toggles the Level 3 command mode. It is used to engage and disengage Command Level 3 as well as specify the Command Drive number.

The CONT and EXEC commands are used to explicitly execute a program or continue the execution of a program. EXEC has a large number of options, including assembling and executing the primary file, loading and executing a disk file, and executing a program stored in memory. CONT is used only to execute a program stored in memory; it is generally used to continue from a breakpoint. This command loads the registers from the register save area and then branches to the program in memory.

CUST and TERM are the customized command and the Terminal subsystem described earlier.

EXIT returns to FDOS.

RESET is the same as a warm start. Refer to Appendix I for a detailed description of the effects of a warm start.

SETC is used to redirect either input, output, or both to a program located in memory.

Finally, TABS is used to set and reset the tab stops used by the input line editor.

2 Primary File Editing

The Primary File Editing commands are <lnum>, APND, DEL, EDIT, FIND, ISRT, and RNUM. These were discussed earlier.

3 Local File Control

The Local File Control commands are FCHK, FILE, LDEL, LDIR, LNAME, LSCR, and RCVR. There were also discussed earlier.

4 Disk File Control

The Disk File Control commands are DDEL, DDIR, and DNAME. They were discussed earlier.

5 Local/Disk File Transfer

The Local/Disk File Transfer commands are LOAD and SAVE. They were discussed earlier.

6 List/Print

The LIST command is used to list all or selected portions of the primary file on the principal I/O device; PRINT prints all or selected portions of the primary file on the printer. If the file is in ARIAN II assembler format, LISTF and PRINF will list and print in columnar format.

7 Program Debugging

The Program Debugging command is BREK. It is used to set, reset, and examine breakpoints in memory. A breakpoint takes the form of a one-byte subroutine call which returns control to ARIAN II, preserving the current values of the registers and the Program Counter.

When a breakpoint is set, the byte of the program addressed by the user is saved in the breakpoint save area and it is replaced by a RST 1 instruction. When breakpoints are reset the replaced byte is copied back to its previous location.

Upon encountering a breakpoint, the breakpoint is automatically reset. The user may then examine the contents of the registers, which is stored in the register save area, and continue program execution by using the CONT command.

8 Assembler

The assembler commands are ASSM and SYMT.

SYMT displays the symbol table from the last assembly on the principal I/O device. The user should employ this command immediately after the assembly; the symbol table is destroyed by disk accesses and the execution of Level 3 commands.

ASSM is used to assemble the primary file. ASSM produces no listing, ASSML produces a normal listing, ASSMP prints the listing on the printer, ASSMF lists the output in formatted mode on the principal I/O device, and ASSMX prints the file in formatted mode on the printer without generating code.

9 Utility

The UTIL command invokes the Utility subsystem; this was discussed earlier.

The ARIAN II User's Manual

References

by
Richard L. Conn

USA Satellite Communications Agency
Fort Monmouth, New Jersey 07703

Conn, Richard. "ARIAN -- An Implementation of a Microcomputer Operating System." May, 1978; MS Thesis, Graduate College, University of Illinois, Urbana, IL 61801. 187 pp. Also available through Defense Documentation Center, Alexandria, VA, under AD Number ADA060210.

Conn, Richard. "ARIAN: A Software Development System for ARIES-1, a Z80-Based Microcomputer." Personal Computing Proceedings, 1979 National Computer Conference, June 4-7, New York, New York; American Federation of Information Processing Societies (AFIPS), AFIPS Press, 210 Summit Avenue, Montvale, New Jersey 07645; p. 363-369.

Conn, Richard. "The Monitor Command System User's Manual." Feb, 1978; Department of Computer Science, University of Illinois, Urbana, IL 61801; Technical Report Number UIUCDCS-R-78-912 and Engineering Report Number UILU-ENG 78 1705. 31 pp.

Conn, Richard. "SDL -- A String Description Language." Project ARIES Report; 24 June 1979; 8 pp.

Conn, Richard. "DEBUG (DBUG) USER'S MANUAL." Project ARIES Report; 1 July 1979; 4 pp.

Conn, Richard. "The USER'S MANUAL for the TEXT FORMATTING SYSTEM." Project ARIES Report; 24 July 1979; 19 pp.

Conn, Richard. "The ARIAN II USER'S MANUAL." Project ARIES Report; 8 July 1979; 150 pp. approximately.

Conn, Richard. "MEMORY TEST USER'S MANUAL." Project ARIES Report; 17 July 1979; 2 pp.

Conn, Richard. "ARIAN DISASSEMBLER USER'S MANUAL." Project ARIES Report; 17 July 1979; 3 pp.

Conn, Richard. "XEDIT USER'S MANUAL." Project ARIES Report; 17 July 1979; 4 pp.

Intel Corp. "8080 Microcomputer Systems User's Manual." Sept, 1975; Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

Intel Corp. "8080 Assembly Language Programming Manual." 1976; Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

Intel Corp. "Interp/80 User's Manual." 1975; Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

Intel Corp. "8008 and 8080 PL/M Programming Manual." 1975; Intel Corporation, 3065 Bowers Avenue, Santa Clara, CA 95051.

Zilog, Inc. "Z80-Assembly Language Programming Manual." 1977; Zilog, 10460 Bubb Road, Cupertino, CA 95014.

MOSTEK. "Z80 Programming Manual." 1977; MOSTEK, 1215 W. Crosby Rd, Carrollton, TX 75006.

North Star Computers, Inc. "The North Star MONITOR." 1977; North Star Computers, 2547 Ninth St, Berkeley, CA 94710.

North Star Computers, Inc. "The North Star Disk Operating System." 1977; North Star Computers, 2547 Ninth St, Berkeley, CA 94710. Version 2, Release 3.

North Star Computers, Inc. "Micro-Disk System MDS-A." 1977; North Star Computers, 2547 Ninth St, Berkeley, CA 94710.

Shugart Associates. "SA400 Minifloppy Diskette Storage Drive." 1977; Shugart Associates, 415 Oakmead Parkway, Sunnyvale, CA 94086. OEM Manual.

Processor Technology Corp. "CUTS User's Manual." 1977; Manual No. 730005; Processor Technology, 7100 Johnson Industrial Way, Pleasanton, CA 94566.

Processor Technology Corp. "ALS-8 User's Manual." 1977; Manual No. 727013; Processor Technology, 7100 Johnson Industrial Way, Pleasanton, CA 94566.

Processor Technology Corp. "SOLOS/CUTER User's Manual." 1977; Processor Technology, 7100 Johnson Industrial Way, Pleasanton, CA 94566.

Digital Research. "An Introduction to CP/M Features and Facilities." 1976; Digital Research, PO Box 579, Pacific Grove, CA 93950.

Digital Research. "CP/M Interface Guide." 1976; Digital Research, PO Box 579, Pacific Grove, CA 93950.

Section 2

Appendix I: The ARIAN II EXECUTIVE COMMANDS

by

Richard L Conn

USA Satellite Communications Agency

Fort Monmouth, New Jersey 07703

APPENDIX I

The ARIAN II EXECUTIVE COMMANDS

Contents

APPENDIX I

The ARIAN II EXECUTIVE Commands

Commands by Functional Group

ARIAN II EXECUTIVE COMMANDS

Introduction

ARIAN II has an extensive set of Executive, or Level 2, commands which give the user a great deal of control over the microcomputer system and the creation of his programs. This control includes the abilities to:

1. Examine and modify memory directly,
2. Control the system environment,
3. Control, modify, and execute all programs available to the user,
4. Debug user programs through specialized system commands, and
5. Assemble programs in ARIAN II Z80 assembly language.

The purpose of this appendix is to catagorize the Executive commands of ARIAN II and explain them in detail. The two contents pages at the beginning of this appendix are designed to be used for quick reference; they index the commannds alphabetically and by function. The descriptions of the commands start on page 6; the commands are ordered alphabetically throughout this appendix.

The entries for each command consist of the following parts:

1. The name of the command,
2. A brief description of the function of the command,
3. A pseudo-SDL representation of the formats the commands can assume,
4. A more detailed description of the function of the command, and
5. Examples of the use of the command.

The following is a String Description Language (SDL) representation and explanation of the "obvious" tokens used in the format representations.

```
<blank> : ' '
<digit> : '0' ! ... ! '9'
<hexdigit> : <digit> ! 'A' ! ... ! 'F'
<alpha> : 'A' ! ... ! 'Z' ! 'a' ! ... ! 'z'
<alphit> : <alpha> ! <digit>
<quote> : '"'
<text> : "any sequence of ASCII characters whose hexadecimal
         values range from 20H to 7EH"
<string1> : <quote> <text>
<string> : <string1> <quote>
<cmndname> : <alpha> <alphit>*3
<fname> : <alpha> <alphit>*7
<lnum> : <digit> <digit>*3
<inc> : <lnum>
<hadr> : <digit> <hexdigit>*3 ! '0' <hexdigit>*4
<delim> : <blank> ! ','
COMMAND_NAME : <cmndname>
FILE_NAME : <fname>
LINE_NUMBER : <lnum>
```

```
HEXVAL : <hadr>
ARIAN_COMMAND : <cmndname> <alphit>* <blank>+
        <fname>? <delim>+
        (<string1> ! <string> <delim>* <string1> !
                <string> <delim>* <string>)
        (<delim>+ (<lnum> ! <hadr>) <delim>*) *3
```

As can be seen by the above description, a command in ARIAN II (ARIAN_COMMAND) can take a large number of forms. It can be thought of as being divided into up to seven free-format fields. These fields are, in order:

1. The name of the ARIAN II command and its affixed options,
2. The name of an ARIAN II file or command [optional],
3. Up to two strings, enclosed in quote marks (") [optional],
4. Up to three line numbers and/or hexadecimal values [optional].

Hence, the following are examples of valid ARIAN II command forms:

```
FILE PROGRAM

LIST 100,2000

LIST 100  2000

TEST  3000  0F0F0  40

TEST2 3000,0F0F0,40

CUSTD TEST

SAVEB1 BINARY1 3400 34FF 06800

FINDF "THIS IS A TEST"

SUBS "THIS IS IT""THIS IS A TEST"

SUBS "THIS IS A TEST", "THIS IS A TEST"

ALLOFIT <fname> "STRING1" "STRING2" 10,20 30

allofit filename "STRING1", "STRING2" 10,20 30
```

The entries in several fields of an ARIAN II command are automatically capitalized internally by the command and argument parser of ARIAN II. Specifically, these fields are the command name and the appended characters (<cmndname> <alphit>*), the file name (<fname>), and the alphabetic hexadecimal digits in the hexadecimal value fields (<hadr>). Hence, the user can type his commands in upper or lower case characters and they will be interpreted in exactly the same way. Also by this token, all command names, options fields,

and file names are forced to consist of capitalized characters by the system. For example, the following forms of commands are equivalent:

Form 1	Form 2
allofit filename "string1"	ALLOFIT FILENAME "string1"
saveb1 binary1 3400 34ff 6800	SAVEB1 BINARY1 3400 34FF 6800
list 100,200	LIST 100 200

In the case of the command "ALLO", the referenced file is named "FILENAME" and the string is "string1". Note that the capitalization rule does not affect strings.

In the case of the command "SAVE", the referenced file is named "BINARY1" and the addresses are 3400, 34FF, and 6800.

In the case of the command "LIST", the lines are 100 and 200.

ARIAN II EXECUTIVE COMMANDS

Command: <lnum>

Brief Description: Any line starting with a line number inserts that line into the primary file at the correct position.

Format: <lnum> <text>

The user may insert a line of text into the primary file or replace a line already in the primary file by simply typing the desired line number followed by a <SP> and the text of the line. This feature of ARIAN provides for very simple line insertion and replacement.

Example: 325 THIS IS LINE 325

Result: The above line (325) is entered into the primary file. If line 325 already exists, it will be replaced by the above line; if it does not exist, line 325 will be inserted between lines 324 and 326 (or the nearest lines to this range) in the primary file.

Command: APND

Brief Description: APND places the following block of lines after the specified or implied line in the file. The block of lines is then typed by the user in block line entry mode.

Format: APND <lnum>?

Format: APNDN <lnum>?

The APND (append) command is one of two block line entry commands of the ARIAN editing subsystem. Block line entry in ARIAN permits the user to type an arbitrary number of lines into the primary file without typing their corresponding line numbers. When the user has finished typing this group of lines, he then types a Ctrl-C followed by a <CR>. These lines will then be entered into the primary file at the appropriate place. If the Ctrl-C is at the end of a line, it will not be included in this line, and this line will be the last line of the block; if Ctrl-C is the first character of a line, the previous line will be the last line of the block.

APND without a line number will enter the block of lines after the last line of the file. APND with a line number will enter the block of lines after the specified line and before the next line in the file.

APND will renumber the file once the block line entry mode is exited, and APNDN will not.

Example: APND

Result: The user is prompted with a "?", after which he types a line of text. If this line is not terminated by a Ctrl-C and <CR>, another prompt appears and he may then type another line of text. This process continues until he either ends a line with a Ctrl-C <CR> or begins a line with a Ctrl-C <CR>. At this time, the block of lines he has just typed will be appended to the primary file and the primary file will be renumbered.

Example: APND 100

Result: The same procedure is executed as described above, but the block of lines is inserted between line 100 and the next line in the file.

Command: ASSM

Brief Description: ASSM causes ARIAN to assemble the primary file.

Format: ASSM <fname>? (<hadr> <hadr>?)?

Format: ASSMF <fname>? (<hadr> <hadr>?)?

Format: ASSML <fname>? (<hadr> <hadr>?)?

Format: ASSMP <fname>? (<hadr> <hadr>?)?

Format: ASSMX <fname>? (<hadr> <hadr>?)?

The ASSM command instructs ARIAN to assemble the primary file. If <fname> is specified, an entry is made in the binary file directory which specifies the boundaries of the object code generated by the assembler. If no address is given, the object code generated by the assembler is placed immediately after the local text file workspace; if one address is given, the object code is placed starting at this address; if two addresses are given, the object code is assembled to execute at the first address and it is physically placed in memory starting at the second.

ASSM assembles the primary file and lists only the lines with errors in them. ASSML lists the file in paged mode on the user's CRT as it is being assembled, ASSMF lists the file in formatted paged mode on the user's CRT as it is being assembled, ASSMP prints the file on the user's printer as it is being assembled, and ASSMX lists the file in formatted paged mode on the user's printer as it is being assembled and does not place any code into memory.

Example: ASSM T1 5800 6800

Result: The primary file will be assembled to execute at location 5800, and the object code will be placed at location 6800. The local binary file T1, which defines the limits of the assembly in terms of the 5800 address will be produced.

Example: ASSML

Result: The primary file is assembled at the end of the workspace. All lines with their object code are listed on the user's CRT. For example, if the workspace is defined to be 2000 to 5000 hexadecimal, the object code will be placed starting at 5001.

Command: BREK

Brief Description: BREK is used to set, reset, and display breakpoints.

Format: BREK <hadr>

Format: BREKD <hadr>

Format: BREKL

Format: BREKS

The breakpoint (BREK) commands permit the user to set, reset, and display the breakpoints. When a breakpoint is reached during a program's execution, the byte replaced by the breakpoint is restored, the contents of the registers is displayed to the user, the contents of all the registers are saved in the register save area of ARIAN, and control is returned to ARIAN. If the user wishes to resume program execution, he may do so by using the CONTinue command.

BREK will set a breakpoint at the specified address; BREKD will clear the breakpoint at the address specified if there is one and inform the user if there is no breakpoint at this address; BREKL will list the addresses of all breakpoints set by the user that have not yet been cleared; and BREKS will scratch (clear) all user breakpoints that have not yet been cleared.

Note: a breakpoint is cleared when it is encountered during a program's execution.

Example: BREK 4000

Result: A one-byte breakpoint is set at location 4000 hexadecimal. If the user's program later encounters this breakpoint, control will be restored to the user through ARIAN.

Example: BREKL

Result: The addresses of all remaining breakpoints will be listed.

Example: BREKS

Result: All remaining breakpoints will be scratched.

Command: CMND

Brief Description: CMND is used to toggle the level 3 command facility and set the number of the CL3 drive.

Format: CMND <cdrive>?

If <cdrive> is specified, the indicated drive number (1-4) will be made the CL3 drive. If it is not specified, CL3 will simply be toggled; if CL3 is on, it will be turned off, and if CL3 is off, it will be turned on with drive 1 being designated the CL3 drive.

The CL3 drive is the drive whose disk is searched for a level 3 command if the search for the current user command fails at levels 1 and 2.

The first digit of the ARIAN II prompt indicates the status of CL3. If this digit is 0, CL3 is off; otherwise, CL3 is on and this digit indicates the number of the CL3 drive.

Example: 01>CMND

Result: 11>
CL3 was disengaged when CMND was typed, and drive 1 was designated to be the CL3 drive as a result.

Example: 11>CMND 4

Result: 41>
Drive 4 was designated the CL3 drive.

Example: 41>CMND

Result: 01>
CL3 was disengaged.

Example: 01>CMND 2

Result: 21>
Drive 2 was designated as the CL3 drive.

Command: CONT

Brief Description: CONT allows the user to continue from a breakpoint or begin execution of a program with specific register values.

Format: CONT <hadr>?

The CONTinue command is one of two commands which allow the user to explicitly execute a program in the microcomputer's memory (EXEC is the other). CONT may be used to either continue a program's execution after a breakpoint has been encountered or explicitly execute a program at a specified address.

When a breakpoint is encountered, the contents of all the registers of the microprocessor are saved in the register save area in the ARIAN scratchpad RAM, thereby permitting the user to examine and modify these values at his leasure through the REGS level 3 command. CONT with no argument will permit the user to continue execution of the program under test with the registers of the microprocessor loaded with the values stored in the register save area.

CONT with a specified address as an argument also loads the registers of the microprocessor, but execution is begun at the address specified. Hence, dynamic debugging of user subroutines is possible by using this command.

Example: CONT 4000

Result: The values stored in the register save area are loaded into the appropriate registers and the program starting at location 4000 hexadecimal is executed.

Command: CUST

Brief Description: CUSTomize allows the user to create, delete, or rename a customized command. Provision is also made to list and scratch all customized commands.

Format: CUST <cname> <hadr>?

Format: CUSTD <cname>

Format: CUSTL

Format: CUSTN <cname>

Format: CUSTS

CUSTomize is perhaps one of the most powerful commands in ARIAN's repertoire. This command allows the user to add his own set of commands to those already executed by ARIAN. The user may give his additional commands any names he wishes, including the names of the commands already defined by ARIAN. If the user wishes to redefine an ARIAN command in this manner, his customized subroutine replaces the system subroutine normally used to execute that command. This replacement is in effect until the user resets ARIAN or deletes his customized command. Other options under the CUST core include CUSTD to delete a specified customized command, CUSTL to list all the customized commands and their execution addresses, CUSTN to rename a customized command, and CUSTS to scratch (delete) all the customized commands.

The CUST command by itself is used to create or redefine a customized commmand. CUST with no address creates a customized command which will begin execution at the default address; CUST with an address defines the command explicitly to execute at the specified address. If a customized command of the same name already exists, the user is prompted with the "REPLACE?" message, to which he responds with a "Y" if he wishes to redefine the execution address of that customized command or "N" if he does not.

The functions of CUSTS, CUSTL, and CUSTD are executed without any particular response to the user. CUSTN prompts the user with "NAME?", to which he may respond with the new name of the file or just a <CR> to abort. The new name is terminated with a <CR>.

Example: CUST TEST

Result: The customized command "TEST" is created. It executes at the current value of the default assembly address.

Example: CUSTL

Result: TEST B800
The names of all the customized commands and their execution addresses are displayed.

Command: DDEL

Brief Description: DDEL allows the user to delete the specified disk file.

Format: DDEL <fname>

Format: DDELn <fname>

DDEL deletes the specified disk file from the disk file directory. Only the directory entry is deleted, but the space occupied by the file is released and subject to user compaction and the disk management routines.

As with all disk-related commands, the disk drive number may be specified as a fifth character of the command.

Example: DDEL TEXT1

Result: The entry for the disk file TEXT1 is deleted from the disk directory.

Example: DDEL3 TEXT1

Result: The entry for the disk file TEXT1 residing on disk drive 3 is deleted from that disk directory. Drive 3 becomes the logged-in drive.

Command: DDIR

Brief Description: DDIR displays an organized listing of the directory of the disk on the specified drive.

Format: DDIR

Format: DDIRn

The disk directory command (DDIR) displays the contents of the directory of the disk currently mounted on the specified drive. Each directory entry contains the name of the file, the length of the file as a decimal number of 256-byte blocks, the type of the file, and the execution or load address of the file if it is a binary file. The types of files permitted by ARIAN are general text files (type 0), binary files (type 1), BASIC program files (type 2), and BASIC data files (type 3).

The disk drive number of the desired drive may be specified as the fifth character of the command. If this is done, that drive automatically becomes the logged-in drive.

Example: DDIR2

Result: The disk directory of the disk on drive 2 is displayed to the user. Drive 2 becomes the logged-in drive.

Command: DEL

Brief Description: DEL allows the user to delete either a specified line or block of lines from the primary file.

Format: DEL <lnum> <lnum>?

The function of the DELete command is to delete lines from the primary file. DEL followed by a line number will delete only that line; DEL followed by two line numbers will delete the block of lines enclosed by the specified lines, inclusive. If a specified line number does not exist in the file, the line number of the line which would follow the specified line if it existed will be used; if the specified line number is larger than the largest line number in the file, the last line will be deleted.

Example: DEL 100

Result: Line 100 is deleted from the primary file.

Example: DEL 100 150

Result: Lines 100 to 150, inclusive, are deleted from the primary file.

Command: DNAME

Brief Description: DNAMe allows the user to rename a specified disk file.

Format: DNAM <fname>

Format: DNAMn <fname>

The disk file rename (DNAM) command renames the specified disk file. In response to this command, ARIAN will prompt the user with "NEW NAME?", to which the user may type the new name for the file or just a <CR> to abort the renaming function.

Like the other disk-related commands, the fifth character of DNAM may be a digit from 1 to 4, specifying the number of the disk drive the file resides on. This drive will be the new logged-in drive as the result.

Example: DNAME OLDFILE

Result: ARIAN will respond with "NEW NAME?", to which the user may respond with the new name of the file OLDFILE. A <CR> will abort the process.

Command: EDIT

Brief Description: EDIT allows the user to edit a specified line of the primary file. It is an intra-line editor.

Format: EDIT <lnum>

The EDIT command invokes the ARIAN intra-line editor. The editor allows the user to edit a line that has already been typed without retyping the entire line. The specified line, or the line that would follow the specified line if this line does not exist, will be edited.

The intra-line editor is a dynamic editor which permits the user to see the effects of his editing commands immediately after he types them. When a line is edited, it is copied into the editor's old line buffer and then displayed to the user. The editor then does a <CR> and prompts the user with a "?". As the user edits this line, each character of the new line that is created is placed into the editor's new line buffer; the original line in the old line buffer is not affected. Finally, when editing is finished, the user may type a <CR> to terminate the editing process and replace the original line in the file with the line as it exists in the new line buffer.

The intra-line editor responds to a number of subcommands. The following is a complete list of these commands and their functions.

1. <BS> -- back up the new line pointer and delete the previous character. This command, echoed as a backspace, is used to delete the previous character in the line. Only the new line pointer is affected by it.
2. <CR> -- terminate creation of the new line. This command terminates editing of the line and replaces the original line in the primary file with the line that currently exists in the new line buffer. If <CR> is the first editing character typed, the edit is aborted and no replacement occurs.
3. <DEL> -- the delete key backs up the new line pointer. The characters backed over are enclosed in "<" and ">" and deleted from the new line. Only the new line pointer is affected by this command.
4. <SP> -- the space bar functions to copy the character pointed to by the old line pointer into the character position pointed to by the new line pointer and advance the old line and new line pointers by one. The space bar, therefore, will simply copy the next character from the old line buffer into the new line buffer. After the copy is done, the copied character will be displayed to the user.
5. A -- abort the editing of the old line. This command may be typed whenever the editor is ready to receive a command (i.e., the editor is not in the middle of an insertion or replacement). It terminates the edit and returns control to ARIAN without affecting the original

line.

6. D -- delete the character pointed to by the old line pointer (delete the next character in the old line). The character is deleted by advancing the old line pointer by one character position and not affecting the new line pointer. The deletion is displayed to the user as a backslash ("\") followed by the deleted character. If the next command typed by the user is another D, the next deleted character is displayed (without the backslash). This will continue until the user types some other command, in which case a closing backslash will be printed and that command will be executed. In effect, the deleted characters will be enclosed in backslashes when displayed to the user.

7. E -- skip to the end of the line. The rest of the characters in the old line buffer are copied into the new line buffer and both pointers are advanced to point to the non-existent character after the last character copied. The copied characters are displayed to the user as they are copied.

8. I -- insert a string of characters in front of the character currently pointed to by the old line pointer. In response to the I typed by the user, the editor types a slash ("/"). The user may they type any string of characters he wishes except for an escape or carriage return. These characters will be copied into the new line buffer, the new line pointer will be advanced, and each character will be echoed to the user as he types it.

The escape and carriage return characters are special characters to the ilsert subcommand. <ESC> instructs the insert subcommand to end the insertion. The editor then types another slash to indicate that the insertion is finished and allows the user to continue editing normally. <CR> instructs the editor to terminate creation of the new line, copy the new line into the primary file, and return to ARIAN command mode. The <CR> is echoed as a slash, a carriage return, and a system prompt, indicating that ARIAN is now in command mode.

The backspace character performs its normal function while in this mode.

9. P -- print the new line and edit it. The command will terminate the new line at the current position of the new line pointer, copy the new line buffer into the old line buffer, print the new line, and restart the editing process with this new line instead of the original line. The original line as it exists in the primary file is not affected.

10. R -- replace the characters pointed to by the old line pointer with the following string. Both pointers are advanced and the new characters are echoed to the user. No special character is typed to the user after he types an R, and the <ESC>, <CR>, and <BS> characters respond as in the I command except that no slash is printed. In effect, as the user types his string, each character he types replaces the corresponding character in the old line buffer.

11. S <letter> -- skip to the specified letter. This is the only two-character command in the editor; it consists of the letter S followed by a single character. When this command is typed, both the old and new line pointers are advanced and the corresponding characters are typed and copied into the new line buffer until the specified character is found or the end of the line is reached. Once the specified character is found, the old line pointer will point to it and this character will not be printed; it will be the next character in the line. The S and the specified letter are not echoed to the user when the command is typed. This command is very useful, particularly when the user wishes to insert, delete, or replace at a specified character; he does not have to space over to that character with this command.

12. X -- exit and reedit the old line. The X command terminates the editing done so far and restarts the edit of the original line. If a P command has been previously typed, the last line placed into the old line buffer is reedited.

The editor has four error messages that it may display. These messages are:

1. ?? -- invalid command. A double question mark indicates that an invalid command has been typed. No recovery is required by the user; he may continue with the desired command.

2. ** -- end of edit line. A double asterisk indicates that the user has tried to go beyond the end of the original line illegally while editing.

3. *EOL* -- end of line buffer. The length of the new line has just reached the limits of the new line buffer, and the user must reedit the original line.

4. <BEL> -- <BS> error. The user has typed a <BS> which attempted to delete a character before the first character in the line.

Example: EDIT 200

Result: Line 200 is printed and the user is prompted with a "?". The user may now edit line 200 using the intra-line editing commands. One useful aspect of this command is that line 200 may be copied as line 201 or any other desired line number by editing only the line number and typing the E (skip to end of line) followed by a <CR>. Line 200 in the primary file will be unchanged and line 201 will be created by this example.

Command: EXEC

Brief Description: EXECute functions to allow the user to execute or assemble and execute a specified or implied file or program.

Format: EXEC (<fname> ! <hadr>)?

Format: EXECn (<fname> ! <hadr>)?

Format: EXECB <fname>?

Format: EXECF <fname>?

Format: EXECL <fname>?

Format: EXECP <fname>?

The EXECute command is one of the most complex of the ARIAN commands. EXEC allows the user to execute a text file, a binary file, or any program or subroutine which resides in memory or on disk. The command also permits a simple, clean return to ARIAN by pushing a return address onto the system stack (which may also be used as the program stack); by keeping the stack stable, the user may return to ARIAN when his program is finished by simply executing a return.

The EXEC command works in many implicit modes. EXEC with no arguments will assemble and execute the primary file. This means of program execution makes software development somewhat easier by permitting the user to execute his primary file, interrrupt the program if it malfunctions, edit the file, and reexecute it with a minimum of effort.

EXEC <fname> will perform the following operations in the order specified:

1. It will first search the local file directory for the specified file. If this file is found, it will be made primary, assembled, and executed.
2. Secondly, if the search in step 1 fails, ARIAN will search the disk directory of the logged-in disk drive for the specified file. If the file is found, it will be loaded into memory.
3. If step 2 fails, an appropriate error message will be given. If step 2 succeeds, a test will be made to see if the file is binary or text. If it is binary, it will be executed; if it is text, it will be assembled and executed.
4. In all cases except when an error occurs, a binary file of the specified name is created.

If EXEC <hadr> is used, the binary program starting at the specified address is executed.

Finally, if EXECB is used, the specified binary file (as defined in the binary file directory) will be executed. If no file name is specified, execution will begin at the default execution address.

The EXECF, EXECL, and EXECP variants perform functions similar to those of

ASSM. EXECF produces a formatted listing on the principal I/O device, EXECL produces a simple listing on the principal I/O device, EXECP produces a simple printing on the printer.

Example: EXEC PROGRAM

Result: If PROGRAM is a local text file, it will be assembled and executed. If PROGRAM is not local and resides on disk, it will be loaded, assembled if it is a text file, and executed.

Example: EXEC 0F000

Result: The machine code beginning at location 0F000 hexadecimal is executed.

Example: EXECB PROGRAMB

Result: The binary file PROGRAMB is executed.

Command: EXIT

Brief Description: EXIT allows the user to transfer control to FDOS.

Format: EXIT

EXIT transfers control to the FDOS disk bootstrap program. FDOS is a modified version of Northstar Corporation's DOS program.

The system status of ARIAN is not affected by this transfer unless the user explicitly or implicitly changes ARIAN or its system RAM area after the control has been transferred.

Reentry may be made to ARIAN in two ways after the transfer has been accomplished: (1) by branching to the ARIAN warm start address (memory location 4) or (2) by branching to the ARIAN cold start address (memory location 0). Reentry at the warm start address preserves the state of ARIAN at the time of the exit.

Command: FCHK

Brief Description: FCHK functions to check the validity of a local text file. This command checks the specified or implied file as to its correctness in format.

Format: FCHK <fname>?

The file check (FCHK) command is used to determine the validity of the specified local text file. This verification includes checking to see that the character count for each line is correct, each line terminates with a <CR>, the file is properly terminated by an <EOF> mark (binary 1), and the local text file directory limits of the file are correct. ARIAN will respond with "VALID FILE" or "INVLD FILE" when the test is finished.

The FILE, RCVR, and LOAD commands do an implicit file check whenever they are executed, but only the invalid message is displayed by their implicit checks.

When a secondary file is checked for validity it is not made primary. There is no overall affect on the ARIAN system as a result of executing this command.

Command: FILE

Brief Description: The FILE command allows the user to explicitly create a new primary local text file or a binary file or view the directory information on the current primary local text file.

Format: FILE (<fname> <hadr>?)?

Format: FILEB <fname> (<hadr> <hadr>)?

The FILE command is one of the most powerful and complex commands in ARIAN. This command is used to create the primary file or a binary file and display the directory entry for the primary file.

The FILE command with no arguments displays the directory entry for the primary file. This entry, like the entries for the secondary local files, consists of the name of the file, the starting and ending memory addresses of the file, and the number of the last line in the file.

The FILE <fname> variant creates a primary file of the name specified. If a local file already exists with this name, it is made primary; the memory manager is invoked, the new primary file is moved to the physical end of the old primary file, and the files are compacted within the currently-defined workspace boundaries. If no local file with this name exists, the new primary file of length zero is created at the end of the last primary file and compaction is again done. Text may now be entered into the primary file by typing line numbers or using the APND command.

If an address is specified with the FILE <fname> variant, the new primary file is placed at this address. Compaction is done to the secondary files, but the new primary file is unaffected by this compaction. Also, the workspace boundary parameters are ignored when the primary file is placed, permitting the user to place this file anywhere he wishes. This FILE variant, then, effectively overrides the memory manager; however, if a later command uses the memory manager, the primary file may be moved and compacted by the memory manager implicitly.

FILEB permits the user to explicitly create a binary file. Binary files are defined solely by their entries in the local binary file directory; they conform to no particular physical structure as they exist in memory. FILEB with no numeric argument creates a binary file with the specified name at the default binary file limits set by the last assembly; FILEB with the two addresses creates the binary file with these addresses as its boundaries.

Example: FILE NEW

Result: If file NEW already exists locally, it is made primary; otherwise, a new text file is created with the name "NEW" at a location determined by the workspace manager. File compaction and workspace compression is then done to all local files.

Example: FILE

Result: The local directory entry for the current primary file is displayed. If this follows the above example and NEW did not previously exist, then an entry like

NEW 3020 3020

would be displayed to the user. This indicates that the current primary file is named NEW, it is a null file, and it begins and ends at location 3020 hexadecimal. If NEW is not null, an entry like

NEW 3020 304C 0050

would be displayed to the user. In this case, NEW resides at locations 3020 to 304C hexadecimal, inclusive.

Example: FILEB BINARY 5000 5010

Result: The local binary file directory entry for the file named BINARY is created. This command defines the file BINARY to reside at locations 5000 to 5010 hexadecimal, inclusive.

Command: FIND

Brief Description: FIND searches the primary file for all occurrances of the specified string.

Format: FIND 'V'? '"' <string> '"'?

Format: FIND 'V'? '"' <string> '"' <lnum>

Format: FINDF 'V'? '"' <string> '"'?

Format: FINDF 'V'? '"' <string> '"' <lnum>

Format: FINDP 'V'? '"' <string> '"'?

Format: FINDP 'V'? '"' <string> '"' <lnum>

The find command performs a search through the primary file for the specified string. If no line number is specified, the search is done over the entire file; if a line number is specified, the search is done starting at the specified line and extending to the end of the file.

The output from the FIND command is a paged listing of the lines which contain the specified string. If the listing is printed on the printer, the output is not paged. If the 'V' option is used, the FIND command will pause after displaying each line. The user may then type <ESC> to abort or anything else to continue.

The FIND command by itself prints the lines as they are currently formatted; FINDF prints the lines in assembler format; and FINDP prints the lines as they are currently formatted on the printer.

Example: FIND "test" 40

Result: All lines after line 40, inclusive, containing the string 'test' will be printed.

Example: FINDF "test

Result: All lines in the file containing the string 'test' (assuming no blanks exist after the word 'test' in the command) will be printed in assembler format.

Command: ISRT

Brief Description: Insert the following block of lines before the specified line in the primary file.

Format: ISRT <lnum>

Format: ISRTN <lnum>

The ISRT command is the same as the APND command, except that the ISRT command places the block of lines in front of the specified line while APND places the block after the specified line. ISRT must have a line number, and it is particularly useful in inserting lines before the first line in the file. APND, on the other hand, is useful for appending lines to the end of the file. ISRT will renumber the file, and ISRTN will not.

Example: ISRT 300

Result: The following block of lines entered in block line entry mode will be placed in front of line 300.

Command: LDEL

Brief Description: LDEL functions to delete the local text file or binary file specified.

Format: LDEL <fname>

Format: LDELB <fname>

The local file delete command (LDEL) is used to delete the directory entry of the specified local binary or text file. This command only deletes the directory entry; the physical file is unaffected by it. LDEL deletes the specified text file, while LDELB deletes the specified binary file.

Command: LDIR

Brief Description: LDIR displays the specified local file directory to the user.

Format: LDIR

Format: LDIRB

The local directory command displays the local text and binary file directories. LDIR displays the local text file directory, and LDIRB displays the local binary file directory. All directory entries consist of the names of the files and their current memory address boundaries. LDIR will display the entry for the current primary file first, and this entry is followed by the entries for the secondary files.

LDIRB displays the local binary file directory and the limits from the last assembly.

Example: LDIR

Result: A list of al the current local text files is displayed to the user. The first file is the primary file.

Example: LDIRB

Result: The local binary file directory is displayed.

Command: LIST

Brief Description: The LIST command is used to list all or selected parts of the primary file on the principal I/O device.

Format: LIST (<lnum> <lnum>?)?

Format: LISTF (<lnum> <lnum>?)?

Format: LISTN (<lnum> <lnum>?)?

The LIST command allows the user to display all or part of the primary file on the pricipal I/O device. All LIST variants are of the same format: if no line number is specified, the entire file is listed; only one line is listed if just one line number is specified; and a block of lines is listed if two lines numbers are given.

LIST displays the requested lines exactly as the user typed them in. LISTF displays the lines in an assembler format in which all labels are aligned in one column, all op codes in another, all operands in a third, and all comments in a fourth. This format assumes that the user entered his lines in an assembler free format in which all labels start in the first assembly column of each line and all op codes not proceeded by a label start in the second assembly column of each line. LISTN displays the requested lines exactly as the user typed them in but without line numbers.

All list displays are paged. This paging varies with the logical I/O device currently assigned as the principal I/O device. At the end of a page, the operator is prompted with "?", to which he may respond with "Y" to continue or "N" to stop. Also, the <ESC> key is monitored throughout the creation of the display, and listing may be terminated at any time by simply hitting this key.

Example: LIST 300 400

Result: Lines 300 to 400, inclusive, are listed on the user's principal I/O device.

Example: LIST 450

Result: Line 450 is listed.

Example: LIST

Result: The entire primary file is listed.

Command: LNAME

Brief Description: This command is used to rename the specified local file.

Format: LNAM <fname>

Format: LNAMB <fname>

The local rename command (LNAMe) allows the user to rename any local binary or text file. LNAM renames a local text file, and LNAMB renames a local binary file. In response to this command, ARIAN prompts the user with "NEW NAME?", to which he may respond with the desired new name for the specified file or a <CR> to abort the renaming process.

Example: LNAM FILE1

Result: ARIAN responds with the prompt "NEW NAME?", to which the user may respond with a valid file name or a <CR>. If he responds with a file name, like F1, FILE1 is renamed F1 in the local text file directory; if he responds with a <CR>, the renaming is aborted.

Command: LOAD

Brief Description: LOAD loads a file from disk into memory.

Format: LOAD <fname> <hadr>?

The LOAD command loads a file from disk into memory. Only binary (type 1) and text (type 2) files may be loaded; an error will be given if the file is of some other type.

If a text file is loaded, the disk directory is searched for the specified file name, and, if found, the local text file memory manager is invoked, the local files are compacted, the specified file is loaded into memory, and the new file is made primary. If a file by the same name already exists locally, the user will be asked if he wishes to replace it by the prompt "REPLACE?"; if the user does not respond with a "N", the file is replaced. If the user specifies an address, the new file is loaded into memory at the specified address, but compaction of the secondary files is still done.

If a binary file is loaded, the disk directory is searched for the specified file name, and, if found, the file is loaded at the execution address given by the disk directory. Any address given in the command becomes the load address. An entry is then made in the local binary file directory.

Example: LOAD LASTF

Result: The disk file LASTF is loaded into memory at a location designated by the local memory manager and made primary if it is a text file. If it is binary, it is loaded at its execution address. In both cases, an entry is made in the appropriate local file directory.

Example: LOAD FILEX 4000

Result: The file FILEX is loaded into memory starting at location 4000 hexadecimal. An entry in the appropriate local file directory is made for it.

Command: LSCR

Brief Description: The LSCR command clears (scratches) the specified local file directory.

Format: LSCR

Format: LSCRB

LSCR scratches (deletes) all entries in the specified local file directory. LSCR scratches the local text file directory, and LSCRB scratches the local binary file directory. The files themselves are not affected by this command -- only the directory entries for them.

Command: PRINT

Brief Description: The PRINT command is the same as the LIST command, but the file is displayed on the printer.

Format: PRIN (<lnum> <lnum>?)?

Format: PRINF (<lnum> <lnum>?)?

Format: PRINN (<lnum> <lnum>?)?

The PRINt command displays the selected line or block of lines on the user's printer. Print displays are not paged. PRIN prints the file as it exists, PRINF prints it in assembler format, and PRINN prints it without line numbers.

The display can be interrupted and control returned to ARIAN by typing the <ESC> key.

If only one line number is given, just that line will be printed. If two line numbers are given, that block of lines, inclusive, will be printed. If no line numbers are given, the entire file will be printed.

Command: RCVR

Brief Description: RCVR allows the user to attempt to recover a local text file whose directory entry has been deleted.

Format: RCVR <fname> <hadr>

The recover (RCVR) command is used to recover a text file that has been deleted from the local text file directory. A validity check is done on the file (see FCHK) starting at the address specified, and a local text file directory entry is created with the specified file name and the file boundaries ascertained by the validity check. If the validity check fails, the recovery is aborted with an appropriate error message. If the validity check succeeds, the new file is made primary implicitly through the FILE command.

Example: RCVR OLDFILE 2001

Result: A validity check is started at location 2001 hexadecimal, and if the validity check succeeds a new local text file of the name OLDFILE is entered into the local text file directory.

Command: RESET

Brief Description: The RESET command resets ARIAN. This is roughly equivalent to a warm start.

Format: RESET

RESEt executes a system reset (warm start) of ARIAN. This system reset includes the following operations:

1. The default execution address is set to the system reset entry point.
2. The assembly limits are reset.
3. The system symbol table is cleared.
4. The system tab stops are reset.
5. The default stack pointer is reset for the CONT command.
6. The principal I/O channel is reset to the default port.
7. Control is returned to ARIAN.

Command: RNUM

Brief Description: The RNUM command is used to renumber the lines in the primary file.

Format: RNUM (<lnum> <inc>?)?

RNUM renumbers the primary file. If no arguments are given, the file is numbered to start at line 5 and continue in increments of 5. If just a line number is specified, the file starts at the specified line number and continues in increments of 5; if both line number and increment are specified, these values are used in the renumbering.

Example: RNUM 100 40

Result: The primary file is renumbered, starting with line 100 and incrementing by 40; i.e., the first line will be 100, and succeeding lines will be 140, 180, 220, etc.

Example: RNUM

Result: The primary file is renumbered, starting at 5 and incrementing by 5 (5, 10, 15, ...).

Command: SAVE

Brief Description: The SAVE command is used to save a file onto disk from memory.

Format: SAVE <fname>

Format: SAVEn <fname>

Format: SAVEB <fname> (<hadr> <hadr> <hadr>?)?

Format: SAVEBn <fname> (<hadr> <hadr> <hadr>?)?

Format: SAVET <fname> <hadr>?

Format: SAVETO <fname>

The SAVE command is used to save text and binary files on disk and change the type and execution address of files currently residing on disk.

The SAVE <fname> variant saves the current primary file on disk under the specified name. This name is not required to be the same as the local name of the primary file. Only the primary file is saved by this command.

The SAVEB variant saves a binary file on disk under the specified name. If no address is given, the default assembly limits are used as the file limits. If two address are given, these are used as the file limits, and if a third address is given, it becomes the execution or load address. If no third address is given, the starting address of the file becomes its execution or load address.

SAVET can be used to change the type of a disk file. SAVETO makes the specified file a type 0 (text) file. SAVET makes the specified file a binary file. If <hadr> is specified, its execution address is set to this value; otherwise, its execution address is set to zero.

An optional fifth character for SAVE and sixth character for SAVEB may be specified. This is a digit from 1 to 4; it specifies the number of the disk drive to save the file on. This drive becomes the new logged-in drive.

Example: SAVE NEW

Result: The local primary text file is saved on disk under the name NEW. Note that this name does not necessarily have to be the same as its local name.

Example: SAVE2 NEWF

Result: The same file is saved under the name NEWF on disk drive 2. Drive 2 is now logged in.

Example: SAVEB1 BIN1 2000 2021 5600

Result: The section of memory from 2000 to 2021 hexadecimal, inclusive, is saved on disk as a binary file with execution address 5600 hexadecimal. It is saved on drive 1, and drive 1 becomes the new logged-in drive.

Example: SAVEB BIN2

Result: The section of memory specified by the default assembly limits is saved on the logged-in drive under the name of BIN2.

Command: SETC

Brief Description: This command allows the user to explicitly redirect I/O within ARIAN.

Format: SETC

Format: SETCI <hadr>

Format: SETCO <hadr>

The SETC command allows the user to redirect all ARIAN system I/O through user-defined I/O routines. The parameter to be passed to and from these routines is passed in the A register. The only constraint placed on these routines is that they do not have a net effect on the system stack or any register and they end with a RET (return) instruction.

SETC by itself resets the system I/O to employ the normal system I/O routines. This is the default upon ARIAN initialization.

SETCI (set customized input) allows the user to redefine the system input routine. All input is routed through the subroutine which begins at the specified address immediately after this command is executed.

SETCO (set customized output) allows the user to redefine the system output routine. All output is routed through the subroutine which begins at the specified address immediately after this command is executed.

Example: SETCI 6C00

Result: All input to ARIAN is directed through the subroutine starting at location 6C00 hexadecimal.

Command: SYMT

Brief Description: The symbol table from the last assembly is displayed to the user on the principal I/O device.

Format: SYMT

Format: SYMT <fname>

The SYMT command displays the symbol table produced in the last assembly if the system has not been reset or the table has not been written over. SYMT displays the user program's symbol table. This is a simple listing of the name of the symbol and its value.

The command 'SYMT' with no arguments simply displays the symbol table as described above. If 'SYMT' is followed by the name of a symbol, which is of the form of <fname>, then that particular symbol will be searched for and its particular value will be displayed if found. For example, 'SYMT DONE' will search the symbol table for the symbol 'DONE' and display its value if found.

Command: TABS

Brief Description: This command allows the user to set, examine, and reset the tab stops.

Format: TABS

Format: TABSL

Format: TABSR

The TABSet command allows the user to set, examine, and reset the ARIAN tab stops. In the set and examine cases, the scale numbering the columns is printed across the page, and the tab stops are denoted by the letter "X".

TABS is used to set the tab stops. After the scale is printed, the user may space or tab (using the previously-defined tab stops) over to the desired tab set location and type an "X". Up to twenty tabs may be set in this way; the process is terminated by typing a carriage return.

TABSL examines (lists) the tab stops as they are currently set. Again, the scale is printed and X's are printed in the columns in which tabs are set.

TABSR resets the tab stops to the standard system definition. Tabs are set at every eighth column.

Command: TERM

Brief Description: This command invokes the terminal subsystem of the UTILITY PROM.

Format: TERM

The TERMinal command invokes the terminal, or inter-system communication, mode of ARIAN. Under this subsystem, the microcomputer becomes relatively transparent to the user and the user's terminal is made to respond like a normal timesharing terminal to an external computer system. Each character typed is sent to a modem and acoustic coupler, and each character received by the modem/coupler is sent to the user's CRT.

The terminal subsystem responds to three subcommands; they are Ctrl-A, Ctrl-L, and Ctrl-R. Ctrl-A invokes the terminal alternate command set; Ctrl-L invokes the download program; Ctrl-R returns control to the calling program (ARIAN).

The terminal alternate command set consists of the following MCS-like commands:

1. F -- disable software echo for full duplex communication.
2. H -- enable software echo for half duplex communication.
3. M -- return to MCS.
4. R <hadr> -- run the subroutine located at the specified address. The return in the subroutine transfers control to the terminal mode (not the terminal alternate command set).
5. T <hadr> -- transfer the downloaded code to the specified address. This command transmits a carriage return (<CR>) to the external computer, and the object code downloaded is loaded into memory starting at the specified address; the addresses given in the code are ignored, and the entire code is loaded sequentially into memory.
6. X -- exit to terminal mode.

Ctrl-L is the download command to the terminal mode subsystem. A carriage return (<CR>) is transmitted to the external computer, and the object code, in INTEL-standard format, is loaded into the microcomputer's memory at the addresses specified in the load blocks.

Ctrl-R simply returns control to the program that called the terminal subsystem. This calling program is usually ARIAN.

Downloading can be interrupted at any time by typing <ESC> on the principal I/O device.

Example: TERM

Result: The terminal subsystem is invoked.

Example: TYPE,OBJECT^L

Result: This the TYPE command to the external computer. OBJECT is an object code file in INTEL-standard format. When the terminal mode reads the Ctrl-L from the user, it transmits a <CR> to the external computer, thereby terminating the TYPE command. The terminal mode then downloads the incoming code into memory and displays it on the CRT.

Command: UTIL

Brief Description: This command invokes the Memory Utility Subsystem.

Format: UTIL

The UTIL command gives the user the ability to examine and modify memory directly. In response to this command, the user is prompted by ARIAN with a slash ("/"). He may then enter the following UTILity subcommands:

1. C <hadr> <hadr> <hadr> -- copy the block of memory defined by the first two addresses to the location of memory starting at the third address. The original block is unchanged unless the third address resides within this block.
2. D <hadr>? <hval>* -- deposit the specified values into memory starting at the given address. If no address is given, the values are deposited starting at the default pointer. When completed, the default pointer points to the next byte to be deposited into.
3. E (<hadr> <hadr>?)? -- examine a specific memory location or a block of memory. The default pointer points to the last location examined. If only one address is given, it and its contents will be displayed followed by a "?", to which the user can respond with a "B" to back up (see the previous address) or "F" to go forward (see the next address). Any other character will exit this mode. If both addresses are specified, this block will be displayed. If no address is specified, the address pointed to by the default pointer will be displayed as though one address was given.
4. F <hadr> <hadr> <hval>* -- find all occurrances of the specified byte string in the memory block bounded by the given addresses.
5. S <hadr>? -- set/display the default pointer. S alone will display the pointer; S with an argument will set the pointer to that value.
6. X -- return to ARIAN command mode.

These commands are a subset of the MCS V1.6 command set, and more details as to their usage may be found in the "MCS User's Manual".

Example: UTIL

Result: The utility subsystem is invoked.

Example: E 0 FF

Result: A block examine is done of locations 0 to FF hexadecimal. The

output is paged.

Example: C 0 FF 4000

Result: Block 0 to FF hexadecimal, inclusive, is copied to start at location 4000 hexadecimal.

Example: D 2000 0 1 2 3

Result: The values 0, 1, 2, and 3 are deposited into memory locations 2000 to 2003 hexadecimal. The default pointer is now pointing to location 2004.

Example: D 4

Result: The value 4 is deposited into memory location 2004 hexadecimal, and the default pointer now points to location 2005 hexadecimal.

Example: X

Result: UTIL is exited and control is returned to ARIAN command mode.

Section 3

Appendix II: The ARIAN II BUFFERS and JUMP TABLE

by

Richard L Conn

USA Satellite Communications Agency

Fort Monmouth, New Jersey 07703

APPENDIX II

The ARIAN II BUFFERS and JUMP TABLE

Contents

The ARIAN II BUFFERS and JUMP TABLE

The ARIAN II Buffers

The following is a description of several of the key buffer areas within the ARIAN II Scratchpad RAM Area. These descriptions include information as to the address of the start of the buffers, the format of the information in the buffers, and the size of the buffers.

Name of Directory/Table: Disk Directory

Entry Format:
16 bytes
- File Name: 8 bytes
- Disk Address (low order, high order): 2 bytes
- Number of Blocks (low order, high order): 2 bytes
- File Type: 1 byte
- Execution Address (low order, high order): 2 bytes
- Unused: 1 byte

Comments:

All elements of an entry except the File Name are in binary. The File Name is in ASCII, with blank fill on the right.

The buffer is large enough to contain 64 entries. The size of the buffer is 1K bytes.

Each entry is of the form described above. If no entry exists at a particular location in the directory, then the first byte of the File Name is a space.

The Starting Address of the Disk Directory is F300.

Name of Directory/Table: Local Text File Directory

Entry Format:
16 bytes
- File Name: 8 bytes
- Beginning of File (BOF) Pointer (low order, high order): 2 bytes
- End of File (EOF) Pointer (low order, high order): 2 bytes
- Maximum Line Number (4 ASCII Characters, MSD First): 4 bytes

Comments:

The File Name is in ASCII, with blank fill on the right. The BOF and EOF

Pointers are in binary, and the Maximum Line Number is in ASCII.

The buffer is large enough to contain 10 entries. The size of the buffer is 160 bytes.

If the first byte of a File Name is a binary 0, then there is no directory entry at this particular location.

The Starting Address of the Local Text File Directory is F090.

Name of Directory/Table: Local Binary File Directory

Entry Format:
12 bytes
File Name: 8 bytes
Starting Address (low order, high order): 2 bytes
Ending Address (low order, high order): 2 bytes

Comments:

The File Name is in ASCII, with blank fill on the right. The Starting and Ending Addresses are in binary.

The buffer is large enough to contain 10 entries. The size of the buffer is 120 bytes.

If the first byte of an entry in the directory is a space, then there is no entry at this particular location.

The Starting Address of the Local Binary File Directory is F018

Name of Directory/Table: Customized Command Table

Entry Format:
6 bytes
Command Name: 4 bytes
Execution Address (low order, high order): 2 bytes

Comments:

The Command Name is in ASCII, with blank fill on the right. The Execution Address is in binary.

The buffer is large enough to contain 20 entries. The size of the buffer is 121 bytes.

The first byte of the table contains a count of the number of entries, and the rest of the table follows the entry format.

The Starting Address of the Customized Command Table is F130.

Name of Directory/Table: Breakpoint Table

Entry Format:
3 bytes
Breakpoint Address (high order, low order): 2 bytes
Replaced Byte: 1 byte

Comments:
All values are in binary.
Note the reversed order of the address.
The buffer is large enough to contain 8 entries. The size of the buffer is 24 bytes.
The presence of an entry in the Breakpoint Table is determined by examining the address at an entry location. If this address is binary 0, then no entry exists here. Note that this prohibits setting a breakpoint at location 0.
The Starting Address of the Breakpoint Table is F1A9.

Name of Directory/Table: Assembler Symbol Table

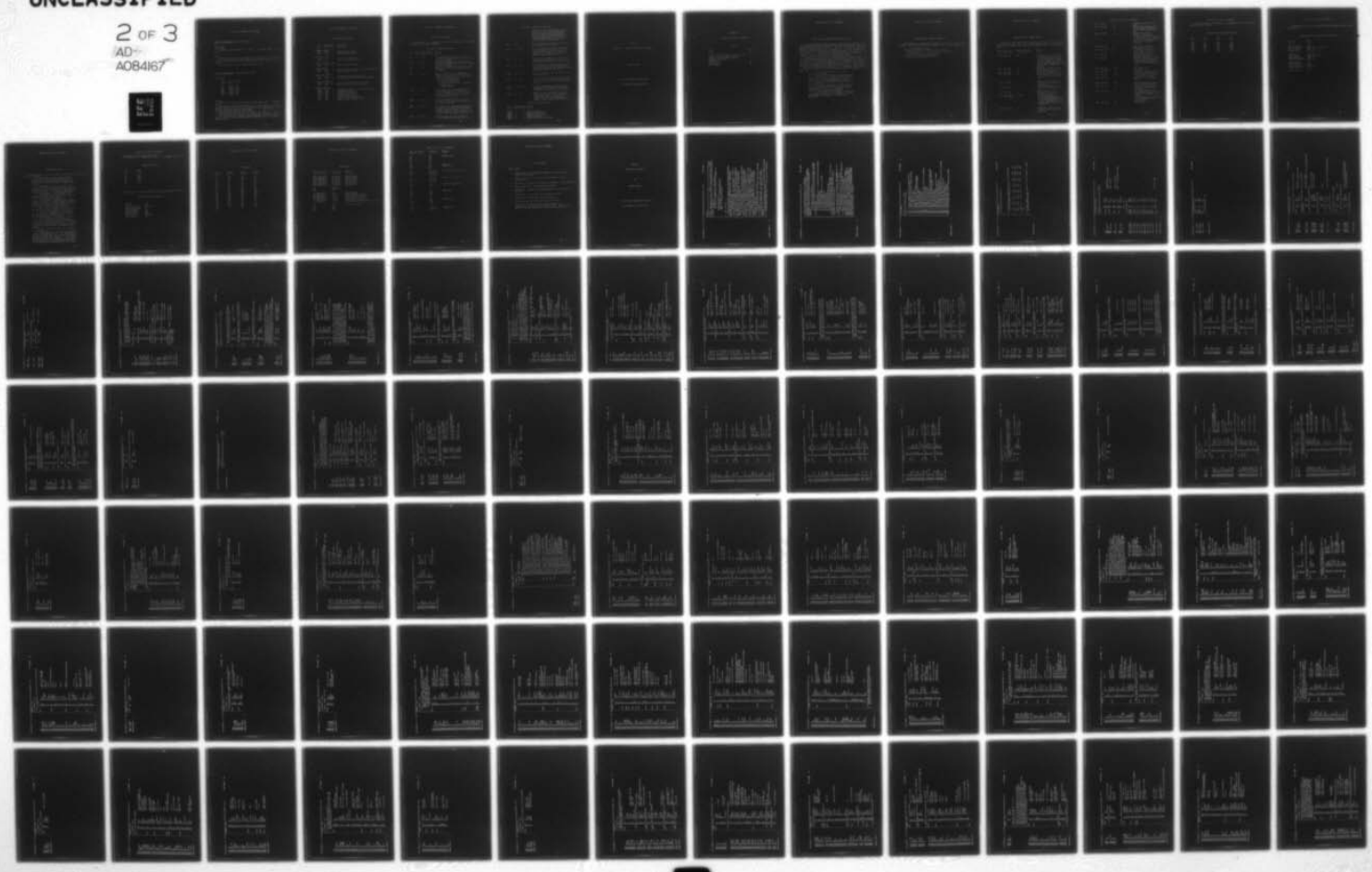
Entry Format:
8 bytes
Symbol Name: 6 bytes
Symbol Address (high order, low order): 2 bytes

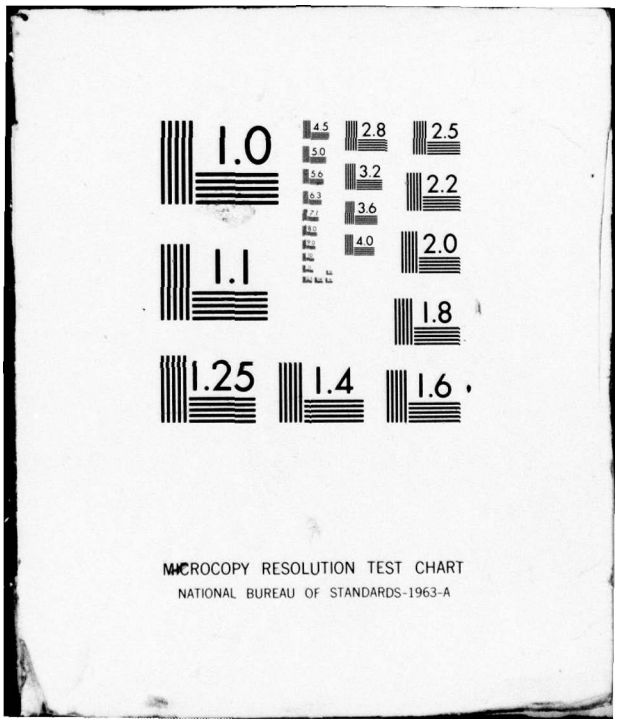
Comments:
The Symbol Name is in ASCII, with zero fill on the right. The Symbol Address is in binary.
The buffer is large enough to contain at least 384 entries. The size of the buffer is flexible, with room for upward growth, but the allocated size is 3172 bytes.
Note the reversed order of the address.
The number of labels in the Assembler Symbol Table is contained in the buffer NOLA (low order, high order). This buffer is 2 bytes long, and its Starting Address is F275.
The Starting Address of the Assembler Symbol Table is F300.

AD-A084 167 ARMY SATELLITE COMMUNICATIONS AGENCY FORT MONMOUTH NJ F/G 9/2
PROJECT ARIES. USER'S MANUALS FOR ARIAN II AND ASSOCIATED SUBSY--ETC(U)
APR 80 R L CONN

UNCLASSIFIED NL

2 OF 3
AD-
A084167




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Name of Directory/Table: Register Save Area

Entry Format:
1 or 2 bytes
The contents of the registers are stored in sequential bytes in this buffer.

Comments:
The sequence of storage of the registers in this area is E, D, C, B, Flags, A, IX (low order, high order), IY (low order, high order), L', H', E', D', C', B', Flags', A', L, H, SP (low order, high order), and PC (low order, high order).
The Starting Address of the Register Save Area is F1C1.

Name of Directory/Table: Executive Parser Buffers

Entry Format:

Buffer	Length	Address
ABUF	16 bytes	F1FD
BBUF	8 bytes	F20D
CBUF	8 bytes	F1F5
FBUF	8 bytes	F1E9
SBUF1	40 bytes	F213
SBUF2	40 bytes	F23B

Comments:
These are the buffers whose contents are filled by the ARIAN II Executive Parser.
ABUF, CBUF, and FBUF are ASCII buffers, with blank fill on the right. SBUF1 and SBUF2 are ASCII buffers; if they contain valid information, the first character in SBUF1 or SBUF2 is a double quote, and the last character is a carriage return. BBUF contains binary values.
ABUF and BBUF may contain up to three entries. For ABUF, each entry is four bytes long; for BBUF, each entry is two bytes long (low order, high order).
The special characters appended to the command name are stored in CBUF in the 5th, 6th, and 7th bytes (SPCHR1, SPCHR2, and SPCHR3, resp.). The address of SPCHR1 is F1F9.

The ARIAN II BUFFERS and JUMP TABLE

Selected ARIAN II Buffers

Name	Address	Size	Use/Function
1. Workspace Buffers			
WSPE	F002	2	Workspace Pointer (End)
WSPS	F000	2	Workspace Pointer (Start)
2. Disk Drive Buffers			
CDRIV	F1DA	1	Command Drive Number Buffer
DRIVEN	F00A	1	Logeed-In Drive Number Buffer
3. Default Execution Address			
EXADR	F00B	2	Default Execution Address
4. Paging Buffers			
LPCNT	F015	1	Line Paging Count (Number of lines left on page)
NL	F265	1	Number of Lines on a Page
5. File Search Buffers			
FEF	F1F4	1	Free Entry Found Flag (use w/XFSEA)
FREAD	F1F2	2	Free Entry in Directory Pointer (use w/XFSEA)
6. Input/Output Buffers			
IBUF	FE00	160	Input Line Buffer (referenced by ZIBF)
OBUF	FDEC	141	Output Line Buffer (used by Assembler)
7. Assembler Buffers			
APND	F1E3	2	Assembler Line Pointer (Pts to current,line)
ASMEN	F26C	2	Assembly End Address
ASMRET	F26F	2	Assembler Return Address
ASMST	F26A	2	Assembly Start Address
ASPC	F268	2	Assembler Location Counter
NOLA	F275	2	Number of Labels in Symbol Table
PASI	F271	1	Assembler Pass Indicator (1 or 2)

The ARIAN II Jump Table

The following is a listing of the entry addresses of the routines accessable via the ARIAN II Jump Table.

The ARIAN II Jump Table Functions

Name	Address	Regs Affected	Function
XCKA11	40	A,HL	Check for the presence of the 1st argument in BBUF/ABUF. Return w/zero set if no argument, ot zero if argument.
XCKA21	43	A,HL	Check for the presence of the 2nd argument in BBUF/ABUF. Return w/zero set if no argument, not zero if argument.
XCKFN1	46	A,HL	Check for the presence of an argument in FBUF. Return w/zero set if no argument, not zero if argument.
XDCOM	69	-All-	Execute the ARIAN II Disk Communication routine; this is the same as FDOS' DCOM, but control is returned to ARIAN II upon error. Input register parameters are: HL = Starting Disk Address DE = Starting RAM Address B = Command (0=write, 1=read, 2=verify) C = Unit Number A = Number of Blocks
XDSCAN	49	-All-	Load and scan directory of Loggen-In Drive for file whose name is in FBUF. Return w/ zero set if found, not zero if not found. If found, HL points to 1st byte of entry in directory for this file.
XFILE	4C	-All-	Execute FILE command. This is useful in creating a new primary file by loading FBUF w th the new file name.
XFIND	4F	-All-	Find a line in the primary file whose number is greater than or equal to the line number (stored as normalized ASCII) in the first element of ABUF. On return, HL points to the 1st byte (character count) of the line found.
XFSEA	5E	-All-	Search the local text file directory for the file whose name is in FBUF. Upon exit,

			return w/not zero & HL pointing to entry in directory if found. If not found, return w/ zero set. Also, if not found, the Free Entry Flag (FEF) is not zero if there is room in the directory for another file and the FREAD buffer points to an available entry location in the directory.
XINK	72	A	Poll all channels for an <ESC> and return to the Executive if one was typed
XLINE	52	-All-	Enter line contained in IBUF into the primary file. Line must be of ARIAN II format. The assembler-defined routine ZLIN creates it properly. A line number (not necessarily normalized) must begin the line.
XLOAD	55	-All-	Execute the LOAD command. This is useful for loading and making primary a disk file on the Logged-In Drive. The file name is in FBUF.
XMESS	5B	-All-	Print string ending in <CR> pointed to by HL on the principal I/O device. Return to the ARIAN II Executive when done. Prefix the string (as printed) w/<CR> '$' and a blank.
XPARSE	6C	-All-	Enter the Executive Parser and parse the line contained in IBUF.
XREAD	6F	HL,A	Enter the input line editor and permit the user to enter a line into IBUF. On exit, HL point to the first char of the line (after the char count).
XRNUM	61	-All-	Execute the RNUM command. The Primary File is renumbered. ABUF contains the non-default parameters as normalized ASCII characters.
XSAVE	58	-All-	Execute the SAVE command. This is useful for creating binary disk files or saving the Primary File (text). Instructions to this routine are passed in the appropriate Executive Parser Buffers, especially SPCHR1.

The ARIAN II Restart Entries

Name	Address	Restart	Function
XARIAN	0	0	ARIAN II Cold Start
XBRKP	8	1	Breakpoint Entry Address
XEOR	66	-	Executive Entry Address
XRESET	4	-	ARIAN II Warm Start
XTRAP	38	7	Memory Trap and Executive Entry

Section 4

Appendix III: SUMMARY of the ARIAN II ASSEMBLER

by

Richard L Conn

USA Satellite Communications Agency

Fort Monmouth, New Jersey 07703

APPENDIX III

SUMMARY of the ARIAN II ASSEMBLER

Contents

SUMMARY of the ARIAN II ASSEMBLER

ARIAN II Standard Assembler Program Line Format

Each line of assembly language source code is divided into from one to four fields. These fields are the name (or label) field, the operation field, the operand field, and the comment field. The name field contains the label assigned to that particular line; this label is a string of alphanumeric characters, the first of which must be alphabetic, and it may optionally end with a colon (:). The operation field contains the mnemonic or pseudo-op which defines the function. The operand field contains the operands required by the instruction, and the comment field is simply descriptive text.

The source line starts with the first assembler line column and extends to the terminating carriage return. This column is the sixth legible character of the line; it is the character following the space which follows the four-digit line number. Hence, an input line consists of a four-digit line number, a space, and the assembly language source code followed by a carriage return. Blank lines are permitted, as are comment lines (lines which are only comment).

The ARIAN II assembler is free format, but a "standard" format for the source code is defined to work with the F and X options on some commands. This format is:

1. The name field must start in the first assembler line column.
2. If the name field exists, the operation field starts one space after the colon or end of the name field. If the name field does not exist, the operation field starts in the second assembler line column.
3. The operand field starts one space after the operation field.
4. The comment field starts one space after the operand field if there is one or one space after the operation field if there is no operand field. The comment field must start with a semicolon.
5. If the entire line is a comment, it must start with either an asterisk (*) or a semicolon (;) in the first assembler line column. The asterisk is preferred.

Assembler Reserved Symbols (Registers)

Certain symbols are reserved by the assembler to refer to the 8080/Z80 registers. They may only be used in the operand field. These symbols are:

1. A -- the accumulator; value 7
2. B, C, D, E, H, and L -- the B, C, D, E, H, and L registers; values are 0, 1, 2, 3, 4, and 5, respectively
3. M -- memory (pointed to by H&L); value 6
4. P, PSW -- the program status word; value 6
5. S, SP -- the stack pointer; value 6

SUMMARY of the ARIAN II ASSEMBLER

Assembler Reserved Symbols (Z-Names)

The ARIAN II assembler also reserves a number of symbolic names, all of which begin with the letter "Z". These symbols are used to reference subroutines and buffers within ARIAN II and the UTILITY PROM. These symbols are:

Symbol	Sample Usage	Registers Affected	Function
ZARG	CALL ZARG	-ALL-	the ARIAN II Executive Parser which parses the line in IBUF as described before; it may be used in conjunction with ZLIN to input a command and parse it for the user's program; ABUF, BBUF, CBUF, S1BUF, and S2BUF are generated
ZBBF	LHLD ZBBF	N/A	the address of BBUF, the binary buffer created by the Parser
ZBLK	CALL ZBLK	A	output <SP> to principal I/O
ZBOF	LHLD ZBOF	N/A	the address of BOFP, the buffer which contains the address of the first byte of the primary file
ZCAH	CALL ZCAH	A	convert the ASCII hexadecimal character in A to its binary equivalent in A
ZCHA	CALL ZCHA	A	convert the low nybble of A to its ASCII hexadecimal equiv in A
ZCR	CALL ZCR	A	output <CR> <LF> to the principal I/O channel
ZCRL	CALL ZCRL DW ADR-$-2	-NONE-	Call Relative Long The two bytes pointed to by the return address make up the relative displacement from the next instruction. The range of displacement is -32768 to +32767.
ZDOT	CALL ZDOT	A	display the A register as up to three decimal digits, left space fill

ZEN	CALL ZEN	A	exchange the nybbles of the A register
ZEOF	LHLD ZEOF	N/A	address of the two-byte buffer which contains the address of the last byte of the primary file
ZEOR	JMP ZEOR	N/A	address of the ARIAN II Executive reentry point -- this entry point can be used to make a clean retur , to ARIAN II and preserve the environment at the time of program execution
ZHOT	CALL ZHOT	A	display the A register as two hexadecimal digits
ZIBF	LXI H,ZIBF	N/A	the address mf the input line buffer IBUF; this buffer is generated by ZLIN
ZIN	CALL ZIN	A	input one character from the principal I/O channel
ZINK	CALL ZINK	A	poll all channels for an <ESC>; if typed, control returned to ARIAN II Executive; return if no character or character not <ESC>
ZJRL	CALL ZJRL DW ADR-$-2	-NONE-	Jump Relative Long (see ZCRL)
ZLIN	CALL ZLIN	A,HL	the ARIAN II input line editor; HL points to IBUF upon exit
ZOUT	CALL ZOUT	-NONE-	output character in A register on principal I/O channel
ZPHL	CALL ZPHL	A	print HL as four hexadecimal digits
ZPPH	CALL ZPPH	A,HL	print string pointed to by HL ending in <CR> on printer
ZPRH	CALL ZPRH	A,HL	print string pointed to by HL ending in <CR> on principal I/O channel
ZPRR	CALL ZPRR	A	print string pointed to by return address ending in <null> (binary 0) on principal I/O channel This string would be specified as "ASC 'string' DB 0" after the call.
ZSHD	CALL ZSHD	BC	BC = HL - DE

SUMMARY of the ARIAN II ASSEMBLER

For quick reference, the following is an alphabetized list of the Assembler Reserved Symbols (Z-Names).

Assembler Reserved Symbols (Z-Names)

Name	Name	Name	Name
ZARG	ZCR	ZHOT	ZOUT
ZBBF	ZCRL	ZIBF	ZPHL
ZBLK	ZDOT	ZIN	ZPPH
ZBOF	ZEN	ZINK	ZPRH
ZCAH	ZEOF	ZJRL	ZPRR
ZCHA	ZEOR	ZLIN	ZSHD

The following is a list of the Assembler Reserved Symbols (Z-Names) grouped by function.

Assembler Reserved Symbols (Z-Names) by Function

Function	Names
Executive Parser	ZARG
Internal Buffers	ZBBF, ZBOF, ZEOF, ZIBF
Specific Output	ZBLK, ZCR
Register Output	ZDOT, ZHOT, ZPHL
Character I/O	ZIN, ZOUT
String Output	ZPPH, ZPRH, ZPRR
Input Line Editor	ZLIN
ASCII/HEX Conversion	ZCAH, ZCHA
Nybble Exchange	ZEN
Interrupt Poll	ZINK
Branch Relative Long	ZCRL, ZJRL
16-bit Arithmetic	ZSHD
ARIAN II Entry	ZEOR

SUMMARY of the ARIAN II ASSEMBLER

Assembler Pseudo Ops

The following is a list and a description of the pseudo-ops recognized by the ARIAN II Assembler.

1. ASC '<string>' -- ASCII string definition. This pseudo-op loads consecutive memory locations starting at the current value of the location counter with the ASCII values of the characters specified in the string.
2. DB <expression> -- define one byte. This instruction evaluates the specified operand and loads one 8-bit value into the memory location pointed to by the location counter. If the number is evaluated into a 16-bit quantity, only the low-order byte is loaded.
3. DS <expression> -- define storage. This reserves the specified number of bytes starting at the current value of the location counter.
4. DW <expression> -- define one word. This instruction evaluates the specified operand, producing a 16-bit value which it loads into memory (low order, high order) at the memory location pointed to by the location counter and the next memory location.
5. END -- end the assembly. This statement is not required; assembly will stop when the end of the file is reached or this statement is encountered.
6. <label> EQU <expression> -- the specified label is assigned the computed value of the operand; the computed value is a 16-bit quantity.
7. EXEC <expression> -- the default execution address is set to the value of the expression.
8. LST -- turn on the listing of the assembly of the program on the principal I/O device if it is not already on. This can be used in conjunction with NLST to list only selected portions of a program during an assembly.
9. NLST -- turn off the listing of the assembly of the program.
10. ORG <expression> -- set the origin (location counter) to the specified value. This instruction also resets the assembly limits and the location in memory at which the object code is loaded. If an ORG appears more than one time in the program, the limits set by the last ORG and the end of the assembly are reflected in the assembly limits displayed by the LDIRB command.

SUMMARY of the ARIAN II ASSEMBLER

All pseudo-ops may be preceeded by a label.
The following is an alphabetized list of the Assembler Pseudo Ops.

Assembler Pseudo Ops

Name	Name
ASC	EQU
DB	EXEC
DS	LST
DW	NLST
END	ORG

The following is a list of the Assembler Pseudo Ops organized by function.

Assembler Pseudo Ops by Function

Function	Name
Constant Definition	ASC, DB, DW
Storage Reservation	DS
Label Assignment	EQU
Location Counter	ORG
Execution Address	EXEC
List Control	LST, NLST
End of Assembly	END

8080 Mnemonics

Mnemonic	Mnemonic	Mnemonic	Mnemonic
ACI	ADC	ADD	ADI
ANA	ANI	CALL	CC
CM	CMA	CMP	CNC
CNZ	CP	CPE	CPI
CPO	CZ	DAA	DAD
DCR	DCX	DI	EI
HLT	IN	INR	INX
JC	JM	JMP	JNC
JNZ	JP	JPE	JPO
JZ	LDA	LDAX	LHLD
LXI	MVI	MOV	NOP
ORA	ORI	OUT	PCHL
POP	PUSH	RAL	RAR
RC	RET	RLC	RM
RNC	RNZ	RP	RPE
RPO	RRC	RST	RZ
SBB	SBI	SHLD	SPHL
STA	STAX	STC	SUB
SUI	XCHG	XRA	XRI
XTHL			

Z80 Mnemonics

Mnemonic & Operand	ZILOG Equiv	Comments
SSPD <expression>	LD (nn),SP	store SP direct
LSPD <expression>	LD SP,(nn)	load SP direct
SBCD <expression>	LD (nn),BC	store BC direct
LBCD <expression>	LD BC,(nn)	load BC direct
SDED <expression>	LD (nn),DE	store DE direct
LDED <expression>	LD DE,(nn)	load DE direct
EXA	EX AF,AF'	
EXX	EXX	
BR <expression>	JR n	branch relative
BC <expression>	JR C,n	branch relative on carry
BNC <expression>	JR NC,n	branch relative on no carry
BZ <expression>	JR Z,n	branch relative on zero
BNZ <expression>	JR NZ,n	branch relative on no zero
DBJ <expression>	DJNZ n	decrement B and branch relative on no zero
LD	LDD	Block Transfer Group
LDR	LDDR	
LI	LDI	
LIR	LDIR	

SUMMARY of the ARIAN II ASSEMBLER

Mnemonic & Operand	ZILOG Equiv	Comments
CD	CPD	Compare Group
CDR	CPDR	
CI	CPI	
CIR	CPIR	
NEG	NEG	Negate A
RLD	RLD	Nybble Rotate
RRD	RRD	
SHB	SBC HL,BC	16-bit subtract with carry
SHD	SBC HL,DE	
SHS	SBC HL,SP	
AHB	ADC HL,BC	16-bit add with carry
AHD	ADC HL,DE	
AHS	ADC HL,SP	
IM0	IM 0	Interrupt Mode Control
IM1	IM 1	
IM2	IM 2	
ID	IND	Input Group
IDR	INDR	
II	INI	
IIR	INIR	
OD	OUTD	Output Group
ODR	OTDR	
OI	OUTI	
OIR	OTIR	
CIN	IN A,(C)	C-Register I/O
COT	OUT (C),A	

Error Messages

Error	Meaning
A	Argument error. The instruction's argument is of the wrong type or generally incorrect.
D	Duplicate label error. The label of this instruction has been used elsewhere.
L	Label error. The label of this instrucion contains an invalid character.
M	Missing label error. A required label is missing.
O	Opcode error. The opcode in the operation field of this instrucion is invalid
R	Register error. The register name is missing or invalid.
S	Syntax error. The instruction syntax is incorrect.
U	Undefined symbol. The referenced symbol is not defined.
V	Value error. The computed value cannot be represented as a 16-bit value or the instruction has a syntax error. Also, an 8-bit value may have been required and a 16-bit value was generated.

Section 5

SOURCE CODE to ARIAN II

by

Richard L Conn

USA Satellite Communications Agency

Fort Monmouth, New Jersey 07703

```
:*ARIANZ2
:*****************************************************************************
:********                        ARIAN                              ********
:******** A MICROCOMPUTER OPERATING AND SOFTWARE DEVELOPMENT SYSTEM ********
:*****************************************************************************
:*  AUTHOR    RICHARD CONN
:*            UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
:*            USA SATELLITE COMMUNICATIONS AGENCY AT FT MONMOUTH, NJ
:*
:*        *** REVISION HISTORY ***
:*          ARIAN, OR ARIAN I -- BEGUN SEPTEMBER, 1977
:*          ARIAN, OR ARIAN I -- COMPLETED MAY, 1978
:*          ARIAN II -- BEGUN APRIL, 1979
:*
:
:***  ARIAN STRUCTURE  ***
:*
:*    ARIAN IS A STRUCTURED, MODULARLY-DESIGNED SYSTEM OF SUBPROGRAMS.
:* THIS STRUCTURED SYSTEM IS DIVIDED INTO FIVE PROGRAM MODULES,
:* FOUR DATA AREAS, AND 1 GENERAL BUFFER REGION.  SPECIFICALLY,
:* THE FIVE PROGRAM MODULES ARE --
:*        1) THE ARIAN JUMP TABLE.  THIS TABLE PROVIDES A NUMBER OF
:* COMMON ENTRY POINTS TO VARIOUS AREAS WITHIN ARIAN.
:*        2) THE INITIALIZE MODULE.  THIS SECTION OF THE PROGRAM HAS
:* TWO ENTRY POINTS -- ONE FOR COMPLETE INITIALIZATION, AND ONE
:* FOR A PARTIAL INITIALIZATION (TO BE USED IN SYSTEM RECOVERY).
:* THESE ENTRY POINTS CAN BE REACHED THROUGH THE 'XARIAN' AND
:* 'XRESET' ENTRIES IN THE JUMP TABLE.
:*        3) THE EXECUTIVE MODULE.  THIS SECTION OF ARIAN IS THE BASIC
:* CONTROL SECTION OF THE PROGRAM.  THE COMMAND PROMPT IS GIVEN,
:* THE INPUT LINE IS READ AND PARSED, AND THE COMMAND IS EXECUTED
:* BY THE EXECUTIVE.  THE EXECUTIVE CAN BE ENTERED EXTERNALLY
:* BY USING THE 'XEOR' ENTRY POINT IN THE JUMP TABLE.
:*        4) THE ARIAN SYSTEM SUPPORT MODULE.  THIS SECTION OF ARIAN
:* PROVIDES A NUMBER OF ROUTINES WHICH ARE OF GENERAL USE
:* BY THE VARIOUS COMMAND LEVEL SUBROUTINES SUPPORTED WITHIN ARIAN.
:*        5) COMMAND LEVEL 2 SUBROUTINE MODULE.  THIS SECTION
:* CONSISTS OF THE RESIDENT SUBROUTINES, SUCH AS THE ASSEMBLER,
:* FILE EDITOR, INTRA-LINE EDITOR, UTILITIES, LINKAGE PROGRAMS,
:* AND OTHER SYSTEM ROUTINES.
:*    THE DATA AREA CONSISTS OF FOUR SECTIONS --
:*        1) THE SYSTEM MESSAGE BLOCK.  ALL PRINTED MESSAGES ARE CONTAINED
:* IN THIS SECTION.
:*        2) THE LEGAL COMMAND TABLE.  THIS IS THE LOOKUP TABLE
:* WHICH CONTAINS THE NAMES AND EXECUTION ADDRESSES OF THE LEVEL 2
:* COMMANDS.
:*        3) THE SYSTEM SYMBOL TABLE.  THIS TABLE CONTAINS THE NAMES AND VALUES
:* OF THE ASSEMBLER'S SYSTEM-DEFINED SYMBOLS.
:*        4) THE ASSEMBLER OPCODE TABLE.  THIS TABLE CONTAINS THE NAMES AND
```

```
:* VALUES OF THE OPCODES RECOGNIZED BY THE ASSEMBLER.
:*    THE BUFFER AREA USED BY ARIAN IS THE THIRD GENERAL
:* SECTION OF ARIAN.  THIS BUFFER AREA, WHICH STATICALLY OCCUPIES
:* THE REGION OF MEMORY FROM 0F000 HEXADECIMAL TO 0F2FF HEXADECIMAL
:* AND DYNAMICALLY OCCUPIES THE AREA BEYOND 0F2FF HEXADECIMAL,
:* CONTAINS --
:*         1) THE WORKSPACE POINTER BUFFERS,
:*         2) THE SPECIAL-PURPOSE BUFFERS USED BY SPECIFIC LEVEL 2 COMMANDS,
:*         3) THE BUFFERS CONTAINING THE NUMBERS OF THE COMMAND DRIVE AND
:* THE LOGGED-IN DRIVE,
:*         4) THE PAGING BUFFERS,
:*         5) THE LOCAL BINARY AND TEXT FILE DIRECTORIES AND RELATED BUFFERS,
:*         6) THE BREAKPOINT BUFFERS,
:*         7) THE REGISTER SAVE AREA,
:*         8) THE SPECIAL-PURPOSE COMMAND LEVEL 3 BUFFERS,
:*         9) THE EDITOR BUFFERS,
:*        10) THE FILE SYSTEM BUFFERS,
:*        11) THE PARSER BUFFERS,
:*        12) THE ASSEMBLER BUFFERS,
:*        13) A RESERVED (SPARE) BUFFER REGION,
:*        14) A BUFFER AREA FOR THE SYSTEM STACK, AND
:*        15) A DYNAMIC BUFFER AREA FOR DISK DIRECTORIES, THE INTRA-LINE
:* EDITOR EDIT BUFFERS, AND THE ASSEMBLER SYMBOL TABLE.
:*    AN EQUATE TABLE FOR THE UTILITY PROM AND MONITOR COMMAND
:* SYSTEM ENTRY POINTS IS ALSO PROVIDED.
:*
:*    ARIAN SUPPORTS THREE LEVELS OF COMMANDS, NAMED COMMAND LEVEL (CL)
:* 1, 2, AND 3.  COMMANDS IN THESE THREE LEVELS ARE EXECUTED
:* BY THE EXECUTIVE AS FOLLOWS --
:*        1) THE EXECUTIVE READS AND PARSES THE INPUT COMMAND,
:*        2) IF THE COMMAND STARTS WITH A DIGIT, THIS LINE IS ENTERED INTO
:* THE PRIMARY FILE; OTHERWISE, IT IS FURTHER PROCESSED.
:*        3) THE FURTHER PROCESSING INVOLVES SCANNING
:* THE CUSTOMIZED COMMAND TABLE (CL1) AND THE EXECUTIVE COMMAND
:* TABLE (CL2), IN THAT ORDER, FOR THE COMMAND NAME.  IF FOUND, THE
:* APPROPRIATE SUBROUTINE IS CALLED; OTHERWISE, A CHECK IS MADE TO
:* SEE IF CL3 IS ENGAGED.  IF CL3 IS ENGAGED, THE COMMAND DRIVE'S
:* DIRECTORY IS LOADED AND SEARCHED FOR THE APPROPRIATE CL3 COMMAND
:* NAME OF THE FORM 'NAME.CMD'.  IF FOUND, THE CORRESPONDING BINARY
:* FILE IS LOADED AND CALLED AS A SUBROUTINE.
:*

PROGRAM ERRORS --      0
```

```
:********                    ********
:******** ARIAN COMMANDS ********
:********                    ********
:
:*  EXEC -- EXECUTE PROGRAM STARTING AT ADDRESS SPECIFIED
:*  UTIL -- INVOKE THE MEMORY UTILITY SUBSYSTEM
:*  FILE -- CREATE MEMORY FILE STARTING AT ADDRESS SPECIFIED
:*  FCHK -- VERIFY SPECIFIED FILE
:*  LIST -- LIST PRIMARY FILE ON CRT
:*  DEL  -- DELETE LINES IN PRIMARY FILE
:*  ASSM -- ASSEMBLE PRIMARY FILE AT ADDRESS SPECIFIED
:*  SYMT -- DISPLAY THE SYMBOL TABLES FROM AN ASSEMBLY
:*  CUST -- DEFINE CUSTOMIZED COMMAND
:*  BREK -- SET/EXAMINE BREAKPOINT
:*  CONT -- CONTINUE FROM BREAKPOINT
:*  RNUM -- RENUMBER PRIMARY FILE
:*  APND -- APPEND TO PRIMARY FILE USING AUTOMATIC LINE NUMBERING
:*  LNAME -- RENAME LOCAL FILE SPECIFIED
:*  DNAME -- RENAME DISK FILE SPECIFIED
:*  DDIR -- GET DISK DIRECTORY OF FILES
:*  LDIR -- GET MEMORY FILE DIRECTORY
:*  LSCR -- SCRATCH (DELETE) DIRECTORY ENTRY OF ALL MEMORY FILES
:*  RCVR -- RECOVER FILE WITH GIVEN NAME AT ADDRESS SPECIFIED
:*  LDEL -- DELETE FILE FROM LOCAL FILE DIRECTORY
:*  TERM -- LINK TO EXTERNAL SYSTEM
:*  EDIT -- INVOKE INTRA-LINE EDITOR
:*  LOAD -- LOAD DISK FILE INTO MEMORY AND MAKE IT PRIMARY
:*  EXIT -- EXIT FROM ARIAN AND INVOKE FDOS
:*  SAVE -- SAVE PRIMARY FILE UNDER NAME SPECIFIED
:*  DDEL -- DELETE SPECIFIED NAME FROM DISK DIR
:*  ISRT -- INSERT A GROUP OF LINES BEFORE THE INDICATED LINE IN AUTO MODE
:*  TABS -- SET, EXAMINE, AND RESET TAB STOPS
:*  RESET -- DO A SYSTEM RESET (RESET THE I/O PORT)
:*  FIND -- FIND A STRING IN THE PRIMARY FILE
:*  PRINT -- PRINT PRIMARY FILE ON PRINTER
:*  SETC -- SET/RESET CUSTOMIZED I/O ADDRESS
:*  CMND -- TOGGLE COMMAND LEVEL 3 AND SET COMMAND DRIVE NUMBER

PROGRAM ERRORS --     0
```

```
                        :**** SEQUENCE OF ARIAN COMMANDS ****
                        :*
                        :*
                        :*
                        :*
                        :*
                        :*  THE EXECUTIVE ARIAN II COMMANDS OCCUR IN THE FOLLOWING ORDER
                        :*    IN THE ARIAN COMMAND EXECUTION SECTION
                        :*
                        :*
                        :*    EXEC    TERM    CUST    LSCR    EXIT    CMND    RCVR    UTIL
                        :*
                        :*    SETC    FIND    EDIT    LOAD    FCHK    DDEL    DNAME   SAVE
                        :*
                        :*    ISRT    APND    LNAME   LDIR    DDIR    LDEL    TABS    FILE
                        :*
                        :*    <LNUM>  LIST    PRINT   RNUM    DEL     SYMT    ASSM    BREK
                        :*
                        :*    CONT
                        :*
                        :*
                        :*  THE 'RESET' COMMAND IS NOT INCLUDED IN THE LIST BECAUSE IT IS SIMPLY
                        :*    AN ENTRY POINT INTO ARIAN.
                        :*
                        :*
```

PROGRAM ERRORS -- 0

```
                        :********                              ********
                        :******** ARIAN CONTROL SECTION ********
                        :********                              ********
                        :
                        :
                        :         ORG     0000H
0000                    :XARIAN   EQU     $
0000  C3 7500           :         JP      ARIAN           ; DEAD START
0003  00                :         NOP                     ;  MONITOR RESTART AT LOCATION 4
0004                    :XRESET   EQU     $
0004  C3 9A00           :         JP      INITA           ; RESET RECOVERY
                        :
                        :         ORG     08H
0008                    :XBRKP    EQU     $
0008  C3 BD1A           :         JP      BRKP            ; BREAKPOINT RESTART
                        :
                        :         ORG     38H             ; RST 7 = INVALID MEMORY ENTERED
0038                    :XTRAP    EQU     $
0038  21 711C           :         LD      HL,MS72
003B  C3 6603           :         JP      MESS
                        :
                        :         ORG     40H
                        :*
                        :*  ARIAN JUMP TABLE
                        :*
0040  C3 AF02           :XCKA11   JP      CKA11
0043  C3 BE02           :XCKA21   JP      CKA21
0046  C3 9F02           :XCKFN1   JP      CKFN1
                        :*
0049  C3 250B           :XDSCAN   JP      DSCAN
                        :*
004C  C3 5C0D           :XFILE    JP      FILE
                        :*
004F  C3 D502           :XFIND    JP      FIND
                        :*
0052  C3 760F           :XLINE    JP      LINE
                        :*
0055  C3 4308           :XLOAD    JP      LOAD
0058  C3 7A09           :XSAVE    JP      DSAVE
                        :*
005B  C3 6603           :XMESS    JP      MESS
                        :*
005E  C3 AF0E           :XFSEA    JP      FSEA
                        :*
0061  C3 2511           :XRNUM    JP      RNUM
                        :*
                        :         ORG     66H             ; NMI ENTRY POINT
0066  C3 BA00           :XEOR     JP      EOR
                        :*

PROGRAM ERRORS --       0
```

```
0069  C3 C302            :XDCOM   JP      DCOM
                         :*
006C  C3 6D01            :XPARSE  JP      PARSE
006F  C3 0101            :XREAD   JP      READ
                         :*
0072  C3 F400            :XINK    JP      INK
                         :*
                         :*  END OF ARIAN JUMP TABLE
                         :*
```

PROGRAM ERRORS -- 0

```
                        :*
                        :*  ARIAN SYSTEM ENTRY POINT -- COMPLETE INITIALIZATION
                        :*
0075                    :ARIAN   EQU     $
0075  CD 5A05           :        CALL    CMNDS           ; INIT CL3
0078  3E01               :        LD      A,1             ; SET DEFAULT LOGGED IN DRIVE
007A  32 0AF0           :        LD      (DRIVEN),A
                        :*
                        :* THIS ROUTINE INITIALIZES THE FILE AREA FOR SUBSEQUENT
                        :*  PROCESSING
                        :*
007D  CD E312           :        CALL    BSCR            ; USE 'BSCR' COMMAND
0080  CD 4C05           :        CALL    SCR1            ; USE 'FSCR' COMMAND & SET A=0 FOR FURTHER INITI
                        :*
                        :*  INITIALIZE WORKSPACE DEFINITION
                        :*
0083  21 002A           :        LD      HL,2A00H        ; START OF WS AT 2A00H
0086  22 00F0           :        LD      (WSPS),HL
0089  21 FFB7           :        LD      HL,0B7FFH       ; END OF WS AT 0B7FFH
008C  22 02F0           :        LD      (WSPE),HL
                        :*
                        :* CLEAR THE BREAKPOINT TABLE
                        :*
008F  21 A9F1           :        LD      HL,BRT
0092  0E18              :        LD      C,NBR*3
0094  CD FA00           :        CALL    CLRZ            : CLEAR
                        :*
                        :*  CLEAR THE CUST TABLE
                        :*
0097  CD 0605           :        CALL    CUSTS           ; CLEAR (SCRATCH) CUSTOMIZED COMMAND TABLE
                        :*
                        :*  CONTINUE INITIALIZATION
                        :*
                        :*
                        :*  SET UP DEFAULT EXECUTION ADDRESS
                        :*
009A                    :INITA   EQU     $
009A  21 9A00           :        LD      HL,INITA        ; ADDRESS
009D  22 0BF0           :        LD      (EXADR),HL
                        :*
                        :*   RESET SYMBOL TABLE AND SYSTEM TABS
                        :*
00A0  21 0000           :        LD      HL,0
00A3  22 75F2           :        LD      (NOLA),HL       ; INITIALIZE NUMBER OF LABELS TO ZERO
00A6  22 D5F1           :        LD      (SPSAV),HL      ; INITIALIZE CONTINUE STACK POINTER
00A9  CD A8D0           :        CALL    TABR            ; SET SYSTEM TAB STOPS
                        :*
                        :*  SET UP PACING LINE COUNT
                        :*

PROGRAM ERRORS --       0
```

```
00AC  3E17              :         LD      A,23            ; 23 LINES/PAGE
00AE  32 65F2           :         LD      (NL),A
                        :*
                        :*  THIS ROUTINE SETS UP THE SIO BOARD
                        :*
00B1  CD 06D0           :         CALL    INIPI           ; SET UP POLLING I/O
                        :*
                        :*    PRINT OPENING MESSAGE
                        :*
00B4  21 621B           :         LD      HL,MSO          ; PRINT '** ARIAN **'
00B7  CD 8A03           :         CALL    SCRN

PROGRAM ERRORS --     0
```

```
                        :*
                        :* THIS IS THE STARTING POINT OF ARIAN.
                        :*    THE STACK IS RESET, THE SYSTEM RETURN ADDRESS IS RESET,
                        :*    THE COMMAND IS INPUT FROM THE USER, COMMENT CHECKING IS
                        :*    DONE, THE COMMAND IS CHECKED AND EXECUTED, AND CONTROL
                        :*    RETURNS TO THIS LOOP
                        :*
00BA                    :EOR    EQU     $
                        :*  RESET EXECUTIVE
00BA  31 00F3           :       LD      SP,STACK        ; RESET STACK
00BD  ED46              :       IM      0               ; SET INTERRUPT MODE 0
00BF  F3                :       DI                      ; DISABLE INTERRUPTS
                        :*  PRINT EXECUTIVE PROMPT
00C0  CD 3FD0           :       CALL    CRLF            ; PRINT CARRIAGE RETURN, LINE FEED, PROMPT
00C3  3A DAF1           :       LD      A,(CDRIV)       ; PRINT COMMAND DRIVE NUMBER
00C6  C630              :       ADD     A,'0'
00C8  CD 1ED0           :       CALL    OUTPUT
00CB  3A 0AF0           :       LD      A,(DRIVEN)      ; PRINT LOGGED DRIVE NUMBER
00CE  C630              :       ADD     A,'0'
00D0  CD 1ED0           :       CALL    OUTPUT
00D3  CD 5DD0           :       CALL    PCHAR           ; PRINT PROMPT
00D6  3E                :       DB      '>'
                        :*  INPUT COMMAND TO EXECUTIVE
00D7  CD 0101           :       CALL    READ            ; READ INPUT LINE
                        :*  EXTRACT FIRST CHARACTER OF COMMAND
00DA  7E                :       LD      A,(HL)          ; FETCH FIRST CHARACTER
                        :*  DETERMINE IF COMMAND IS A COMMENT AND PROCESS IF SO
00DB  FE30              :       CP      '0'             ; '0'?
00DD  38DB              :       JR      C,EOR-$         ; COMMENT IF LESS THAN '0'
00DF  FE3B              :       CP      ';'             ; COMMENT IF ';'
00E1  28D7              :       JR      Z,EOR-$
                        :*  DETERMINE IF LINE NUMBER AND PROCESS IF SO
00E3  FE3A              :       CP      '9'+1           ; COMMAND OR LINE NUMBER?
00E5  3005              :       JR      NC,EORP-$
00E7  CD 760F           :       CALL    LINE            ; PROCESS AS LINE NUMBER
00EA  18CE              :       JR      EOR-$
                        :*  PARSE COMMAND
00EC  CD 6601           :EORP   CALL    VALC            ; GET COMMAND VALUES
                        :*  SCAN FOR AND EXECUTE COMMAND
00EF  CD 0B01           :       CALL    COMM            ; CHECK LEGAL COMMANDS
                        :*  CONTINUE EXECUTIVE
00F2  18C6              :       JR      EOR-$

PROGRAM ERRORS --    0
```

```
                        :********                                    ********
                        :******** ARIAN SUPPORT ROUTINE SECTION ********
                        :********                                    ********
                        :
                        :
                        :*
                        :*  INK -- INTERRUPT CHECK; THIS ROUTINE QUERIES THE
                        :*    INPUT PORT FOR AN <ESC>; IF ONE WAS TYPED, CONTROL
                        :*    IS TRANSFERRED TO EOR ELSE A RETURN IS DONE
                        :*
00F4                    :INK     EQU     $
00F4  CD 78D0           :        CALL    INK1            ; CHECK WITH SYSTEM INK
00F7  28C1              :        JR      Z,EOR-$         ; ABORT
00F9  C9                :        RET
                        :*
                        :*  THIS ROUTINE ZEROES OUT B&C BYTES OF MEMORY STARTING AT
                        :*     THE ADDRESS POINTED TO BY H&L; THE CONSTANT IS IN A
                        :*
00FA                    :CLRZ    EQU     $
00FA  AF                :        XOR     A               ; A=0 FOR CLRA
00FB                    :CLRA    EQU     $
00FB  77                :        LD      (HL),A          ; STORE CONST A
00FC  23                :        INC     HL              ; PT TO NEXT
00FD  0D                :        DEC     C               ; DECR COUNT
00FE  20FB              :        JR      NZ,CLRA-$
0100  C9                :        RET
                        :*
                        :*  THIS ROUTINE INTERFACES TO THE ARIES-1 POLLING
                        :*       READ UTILITY AND SETS THE ADDRESS POINTED TO BY ADDS
                        :*       TO IBUF
                        :*
0101                    :READ    EQU     $
0101  CD AED0           :        CALL    READ1           ; READ LINE
0104  21 01FE           :        LD      HL,IBUF         ; SET ADDRESS POINTER
0107  22 E5F1           :        LD      (ADDS),HL       ; STORE POINTER
010A  C9                :        RET
                        :*
                        :* THIS ROUTINE CHECKS THE INPUT COMMAND AGAINST ALL
                        :*  LEGAL COMMANDS STORED IN A TABLE.  IF A LEGAL COMMAND
                        :*  IS FOUND, A JUMP IS MADE TO THAT ROUTINE.  OTHERWISE
                        :*  AN ERROR MESSAGE IS OUTPUT TO THE USER.
                        :* WHEN TRANSFER IS MADE TO THE LOCATED LEVEL 1 OR LEVEL 2
                        :*  ROUTINE, THE A REGISTER CONTAINS THE VALUE OF SPCHAR.
                        :*  THIS IS FOR THE CONVENIENCE OF THE ARIAN II SYSTEM PROGRAMMER
                        :*
010B                    :COMM    EQU     $
010B  11 30F1           :        LD      DE,CUSTT        ; CUSTOMIZED COMMAND TABLE
010E  3E04              :        LD      A,4             ; LENGTH OF COMMAND
0110  32 72F2           :        LD      (NCHR),A        ; STORE IN RESERVED SPACE
```

PROGRAM ERRORS -- 0

```
0113  1A              :        LD      A,(DE)          ; GET NUMBER OF COMMANDS
0114  47              :        LD      B,A             ; STORE IN B
0115  B7              :        OR      A               ; NO COMMAND?
0116  2806            :        JR      Z,COMM0-$
0118  13              :        INC     DE              ; POINT TO START OF TABLE
0119  CD 2D01         :        CALL    COMS            ; SEARCH
011C  280B            :        JR      Z,COMM1-$
011E  11 991C         :COMM0   LD      DE,CTAB         ; COMMAND TABLE ADDRESS
0121  0621            :        LD      B,NCOM          ; NUMBER OF COMMANDS
0123  CD 2D01         :        CALL    COMS            ; SEARCH TABLE
0126  C2 8305         :        JP      NZ,CMND1        ; CHECK CL3
0129  3A F9F1         :COMM1   LD      A,(SPCHAR)      ; A=(SPCHAR) FOR EXEC ROUTINES
012C  E9              :        JP      (HL)
                      :*
                      :* THIS ROUTINE CHECKS TO SEE IF A BASE CHARACTER STRING
                      :*  IS EQUAL TO ANY OF THE STRINGS CONTAINED IN A TABLE
                      :*  POINTED TO BY D,E.  THE TABLE CONSISTS OF ANY NUMBER
                      :*  OF CHARS, WITH 2 BYTES CONTAINING VALUES ASSOCIATED
                      :*  WITH IT.  REG B CONTAINS THE NO OF STRINGS TO COMPARE.
                      :*  THIS ROUTINE CAN BE USED TO SEARCH THROUGH A COMMAND
                      :*  OR SYMBOL TABLE.  ON RETURN, IF THE ZERO FLAG IS SET,
                      :*  A MATCH WAS FOUND; IF NOT, NO MATCH WAS FOUND.  IF
                      :*  A MATCH WAS FOUND, D,E POINT TO THE LAST BYTE
                      :*  ASSOCIATED WITH THE CHARACTER STRING.  IF NOT, D,E
                      :*  POINT TO THE NEXT LOCATION AFTER THE END OF THE TABLE.
                      :*
012D                  :COMS    EQU     $
012D  2A E5F1         :        LD      HL,(ADDS)       ; FETCH COMPARE ADDRESS
0130  3A 72F2         :        LD      A,(NCHR)        ; GET LENGTH OF STRING
0133  4F              :        LD      C,A
0134  CD 4201         :        CALL    SEAR            ; COMPARE STRINGS
0137  1A              :        LD      A,(DE)          ; FETCH VALUE
0138  6F              :        LD      L,A
0139  13              :        INC     DE
013A  1A              :        LD      A,(DE)          ; FETCH VALUE
013B  67              :        LD      H,A
013C  C8              :        RET     Z
013D  13              :        INC     DE              ; SET TO NEXT STRING
013E  10ED            :        DJNZ    COMS-$
0140  04              :        INC     B               ; CLEAR ZERO FLAG
0141  C9              :        RET
                      :*
                      :* THIS ROUTINE CHECKS TO SEE IF TWO CHARACTER STRINGS IN
                      :*  MEMORY ARE EQUAL.  THE STRINGS ARE POINTED TO BY D,E
                      :*  AND H,L.  ON RETURN, THE ZERO FLAG SET INDICATES A
                      :*  MATCH.  REG C INDICATES THE LENGTH OF THE STRINGS.  ON
                      :*  RETURN, THE POINTERS POINT TO THE NEXT ADDRESS AFTER
                      :*  THE CHARACTER STRINGS.
                      :*

PROGRAM ERRORS --     0
```

```
0142                    :SEAR    EQU     $
0142  1A                :        LD      A,(DE)          ; FETCH CHARACTER
0143  BE                :        CP      (HL)            ; COMPARE CHARACTERS
0144  200A              :        JR      NZ,SEAR1-$
0146  FE21              :        CP      ' '+1           ; MATCH IF <SP> OR LESS
0148  380B              :        JR      C,SEAR2-$
014A  23                :        INC     HL
014B  13                :        INC     DE
014C  0D                :        DEC     C               ; DECREMENT CHARACTER COUNT
014D  20F3              :        JR      NZ,SEAR-$
014F  C9                :        RET
0150  CD 5501           :SEAR1   CALL    SEAR2           ; SKIP OVER REST
0153  0C                :        INC     C               ; CLEAR ZERO FLAG
0154  C9                :        RET
0155  13                :SEAR2   INC     DE              ; SKIP OVER REST OF ENTRY
0156  23                :        INC     HL
0157  0D                :        DEC     C
0158  20FB              :        JR      NZ,SEAR2-$
015A  C9                :        RET                     ; RETURN W/ZERO FLAG SET
                        :*
                        :* THIS ROUTINE ZEROES OUT A BUFFER IN MEMORY WHICH IS
                        :*  THEN USED BY OTHER SCANNING ROUTINES.
                        :*
015B                    :ZBUF    EQU     $
015B  AF                :        XOR     A               ; GET A ZERO
015C  11 0DF2           :        LD      DE,ABUF+16      ; BUFFER ADDRESS
015F  0610              :        LD      B,16            ; BUFFER LENGTH
0161  1B                :ZBUF1   DEC     DE              ; DECREMENT ADDRESS
0162  12                :        LD      (DE),A          ; ZERO BUFFER
0163  10FC              :        DJNZ    ZBUF1-$
0165  C9                :        RET
                        :*
                        :* THIS ROUTINE CALLS PARSE TO OBTAIN THE INPUT PARAMETER
                        :*  VALUES AND CALLS AN ERROR ROUTINE IF AN ERROR OCCURRED
                        :*  IN THAT ROUTINE.
                        :*
0166                    :VALC    EQU     $
0166  CD 6D01           :        CALL    PARSE           ; GET INPUT PARAMETERS
0169  D0                :        RET     NC
016A  C3 F903           :        JP      PARAME          ; 'PARAMETER ERROR'
                        :*
                        :* THIS ROUTINE EXTRACTS THE VALUES ASSOCIATED WITH A
                        :*  COMMAND FROM THE INPUT STREAM AND PLACES THEM IN THE
                        :*  ASCII BUFFER (ABUF).  IT ALSO CALLS A ROUTINE TO
                        :*  CONVERT THE ASCII HEXADECIMALS TO BINARY AND STORES
                        :*  THEM IN THE BINARY BUFFER (BBUF).  ON RETURN, CARRY
                        :*  SET INDICATES AN ERROR IN INPUT PARAMETERS.
                        :*
                        :* THE COMMAND LINE IS DIVIDED INTO THE FOLLOWING FIELDS
```

```
                        :*    1. COMMAND NAME (4 CHARS)
                        :*    2. SPECIAL CHARACTERS (ONE TO THREE, 1 CHAR EA)
                        :*    3. FILE NAME (8 CHARS)
                        :*    4. STRINGS 1&2 (40 CHARS EA)
                        :*    5. NUMERIC ARGUMENTS (ONE TO THREE, 4-5 CHARS EA)
                        :*
                        :* DELIMITERS OF EITHER <BLANK> OR COMMA ARE REQUIRED BETWEEN EACH
                        :*  FIELD EXCEPT STRINGS 1&2, IN WHICH CASE THE TERMINATION OF THE
                        :*  STRINGS IS ADEQUATE.  A LITERAL CHARACTER ESCAPE IS PROVIDED SO THE
                        :*  USER MAY SPECIFY THE DOUBLE QUOTE AS A CHARACTER WITHIN A STRING.
                        :*  THE BACKSLASH TELLS THE PARSER TO INTERPRET THE NEXT CHAR LITERALLY,
                        :*  SO BACKSLASH FOLLOWED BY A DOUBLE QUOTE GIVES THE DOUBLE QUOTE CHAR AND
                        :*  BACKSLASH FOLLOWED BY A BACKSLASH GIVES THE BACKSLASH CHAR.
                        :*
016D                    :PARSE   EQU     $
                        :*  CLEAR PARSER BUFFERS -- FBUF, CBUF, ABUF, BBUF, S1BUF, S2BUF
016D  21 E9F1           :        LD      HL,FBUF          ; ZERO FBUF, CBUF, ABUF, BBUF, S1BUF, S2BUF
0170  0E7A              :        LD      C,122            ; 122 BYTES
0172  CD FA00           :        CALL    CLRZ             ; STORE ZEROES
                        :*  POINT TO INPUT LINE
0175  21 00FE           :        LD      HL,IBUF-1
0178  E5                :        PUSH    HL               ; SAVE HL
                        :*  EXTRACT COMMAND AND PLACE IT IN CBUF
0179  0607              :        LD      B,7              ; 7 CHARS IN COMMAND NAME MAX
017B  11 F5F1           :        LD      DE,CBUF          ; POINT TO COMMAND NAME BUFFER
017E  23                :PAR1    INC     HL               ; PT TO NEXT CHAR
017F  7E                :        LD      A,(HL)           ; GET CHAR
0180  FE21              :        CP      ' '+1            ; DONE?
0182  3808              :        JR      C,PAR2-$
0184  CD 05D1           :        CALL    CAPS             ; CONVERT LOWER CASE TO UPPER CASE
0187  77                :        LD      (HL),A           ; STORE CONVERTED CHARACTER
0188  12                :        LD      (DE),A           ; STORE CHAR IN CBUF
0189  13                :        INC     DE               ; PT TO NEXT CHAR IN CBUF
018A  10F2              :        DJNZ    PAR1-$
018C  E1                :PAR2    POP     HL               ; RESTORE HL
                        :*  SCAN OVER FIRST ELEMENT IN LINE & RETURN IF EOL
018D  23                :PAR3    INC     HL               ; PT TO NEXT CHAR
018E  7E                :        LD      A,(HL)           ; FETCH INPUT CHARACTER
018F  FE20              :        CP      ' '              ; LOOK FOR FIRST BLANK CHARACTER
0191  3F                :        CCF
0192  D0                :        RET     NC
0193  20F8              :        JR      NZ,PAR3-$        ; JUMP IF NO BACK
                        :*  SCAN TO NEXT PARAMETER
0195  22 73F2           :        LD      (PNTR),HL        ; SAVE POINTER
0198  CD 7903           :        CALL    SBLK             ; SCAN TO FIRST PARAMETER
019B  3F                :        CCF
019C  D0                :        RET     NC               ; RETURN IF CR
                        :*  CHECK FOR PRESENCE OF FILE NAME
019D  FE3F              :        CP      '?'

PROGRAM ERRORS --     0
```

```
019F  3829            :        JR      C,PAR8-$        ; NO FILE NAME
                      :*  PLACE FILE NAME IN FBUF
01A1  11 E9F1         :        LD      DE,FBUF         ; NAME FOLLOWS PUT IN FBUF
01A4  0E08            :        LD      C,NMLEN
01A6  2B              :        DEC     HL
01A7  23              :PAR4    INC     HL
01A8  7E              :        LD      A,(HL)
01A9  FE21            :        CP      ' '+1           ; <CR>, <SP>=END OF FILE NAME
01AB  380E            :        JR      C,PAR5-$
01AD  FE2C            :        CP      ','             ; CHECK FOR COMMA AS TERMINATOR
01AF  280A            :        JR      Z,PAR5-$
01B1  CD 05D1         :        CALL    CAPS            ; CONVERT LOWER CASE TO UPPER CASE
01B4  12              :        LD      (DE),A          ; STORE FILE NAME
01B5  13              :        INC     DE
01B6  0D              :        DEC     C               ; CHECK CHAR COUNT
01B7  280C            :        JR      Z,PAR7-$
01B9  18EC            :        JR      PAR4-$
                      :*  SPACE FILE END OF FILE NAME
01BB  3E20            :PAR5    LD      A,' '           ; GET AN ASCII SPACE
01BD  0D              :PAR6    DEC     C
01BE  FA C501         :        JP      M,PAR7
01C1  12              :        LD      (DE),A          ; FILL IN WITH SPACES
01C2  13              :        INC     DE
01C3  18F8            :        JR      PAR6-$
                      :*  SCAN TO NEXT PARAMETER AND RETURN IF EOL
01C5  CD 8403         :PAR7    CALL    SBLK2
01C8  3F              :        CCF
01C9  D0              :        RET     NC
                      :*  CHECK FOR STRING PARAMETER
01CA  FE22            :PAR8    CP      '"'             ; STRING?
01CC  2012            :        JR      NZ,PAR9-$
                      :*  STORE FIRST STRING PARAMETER IN S1BUF
01CE  11 13F2         :        LD      DE,S1BUF        ; STRING BUFFER
01D1  CD 3902         :        CALL    PAR12           ; STORE STRING CHARS
01D4  D0              :        RET     NC
                      :*  CHECK FOR SECOND STRING PARAMETER
01D5  FE22            :        CP      '"'             ; 2ND STRING?
01D7  2007            :        JR      NZ,PAR9-$
                      :*  STORE SECOND STRING PARAMETER IN S2BUF
01D9  11 3BF2         :        LD      DE,S2BUF        ; 2ND STRING BUFFER
01DC  CD 3902         :        CALL    PAR12           ; STORE STRING CHARS
01DF  D0              :        RET     NC
                      :*  EXTRACT AND STORE NUMERIC PARAMETERS IN ABUF AND BBUF
01E0  11 FDF1         :PAR9    LD      DE,ABUF
                      :*  FIRST NUMERIC PARAMETER
01E3  CD 2902         :        CALL    PAR10           ; PLACE PARAMETER IN BUFFER AND CHECK DIGIT COUN
01E6  D8              :        RET     C               ; RETURN IF TOO MANY DIGITS
01E7  01 FDF1         :        LD      BC,ABUF
01EA  CD 6F02         :        CALL    AHEX            ; CONVERT VALUE
```

PROGRAM ERRORS -- 0

```
01ED  D8              :        RET     C               ; ILLEGAL CHARACTER
01EE  22 0DF2         :        LD      (BBUF),HL       ; SAVE IN BINARY BUFFER
01F1  21 FDF1         :        LD      HL,ABUF
01F4  CD 3102         :        CALL    PAR11           ; NORMALIZE AND SCAN TO NEXT PARM
01F7  D0              :        RET     NC              ; RETURN IF CR
01F8  11 01F2         :        LD      DE,ABUF+4
                      :*  SECOND NUMERIC PARAMETER
01FB  CD 2902         :        CALL    PAR10           ; PLACE PARM IN BUFFER AND CHECK DIGIT COUNT
01FE  D8              :        RET     C               ; RETURN IF TOO MANY DIGITS
01FF  01 01F2         :        LD      BC,ABUF+4
0202  CD 6F02         :        CALL    AHEX            ; CONVERT VALUE
0205  D8              :        RET     C               ; ILLEGAL VALUE
0206  22 0FF2         :        LD      (BBUF+2),HL     ; SAVE IN BINARY BUFFER
0209  21 01F2         :        LD      HL,ABUF+4
020C  CD 3102         :        CALL    PAR11           ; NORMALIZE AND SCAN TO NEXT PARM
020F  D0              :        RET     NC              ; RETURN IF CR
0210  11 05F2         :        LD      DE,ABUF+8
                      :*  THIRD NUMERIC PARAMETER
0213  CD 2902         :        CALL    PAR10           ; PLACE PARM IN BUFFER AND CHECK DIGIT COUNT
0216  D8              :        RET     C               ; RETURN IF TOO MANY
0217  01 05F2         :        LD      BC,ABUF+8
021A  CD 6F02         :        CALL    AHEX            ; CONVERT VALUE
021D  D8              :        RET     C               ; ILLEGAL VALUE
021E  22 11F2         :        LD      (BBUF+4),HL     ; SAVE IN BINARY BUFFER
0221  21 05F2         :        LD      HL,ABUF+8
0224  CD 2D03         :        CALL    NORM            ; NORMALIZE ASCII VALUE
0227  B7              :        OR      A               ; CLEAR CARRY
0228  C9              :        RET
                      :*  EXTRACT NUMERIC PARAMETER AND PLACE IN CORRESPONDING ABUF
0229  CD 8B18         :PAR10   CALL    ALPS            ; PLACE PARAMETER IN BUFFER
022C  78              :        LD      A,B             ; CHECK FOR 1-5 DIGITS
022D  FE06            :        CP      6
022F  3F              :        CCF
0230  C9              :        RET
                      :*  NORMALIZE NUMERIC PARAMETER AND PLACE IN CORRESPONDING BBUF
0231  CD 2D03         :PAR11   CALL    NORM            ; NORMALIZE
0234  CD 7903         :        CALL    SBLK            ; SCAN TO NEXT PARAMETER
0237  3F              :        CCF
0238  C9              :        RET
                      :*  EXTRACT AND STORE STRINGS
0239  12              :PAR12   LD      (DE),A          ; STORE CHAR
023A  23              :        INC     HL              ; PT TO NEXT
023B  13              :        INC     DE
023C  7E              :        LD      A,(HL)          ; GET CHAR
023D  FE0D            :        CP      0DH
023F  280D            :        JR      Z,PAR14-$
0241  FE5C            :        CP      '\'             ; ESCAPE (LITERAL) CHARACTER
0243  2004            :        JR      NZ,PAR13-$
0245  23              :        INC     HL              ; PT TO LITERAL CHARACTER

PROGRAM ERRORS --     0
```

```
0246  7E              :        LD      A,(HL)          ; LOAD LITERAL INTO A FOR STORAGE
0247  18F0            :        JR      PAR12-$
0249  FE22            :PAR13   CP      '"'             ; END OF STRING?
024B  20EC            :        JR      NZ,PAR12-$
024D  23              :        INC     HL              ; PT TO CHAR BEYOND END OF STRING
024E  3E0D            :PAR14   LD      A,0DH           ; ENDING <CR> STORED AT END OF STRING
0250  12              :        LD      (DE),A
0251  CD 7C03         :        CALL    SBLK1           ; SKIP TO NEXT CHAR
0254  3F              :        CCF
0255  C9              :        RET
                      :*
                      :* THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                      :*  BY B,C AND CONVERTS THE ASCII DECIMAL DIGITS INTO
                      :*  BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                      :*  SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                      :*
0256                  :ADEC    EQU     $
0256  21 0000         :        LD      HL,0            ; GET A 16-BIT ZERO
0259  0A              :ADEC1   LD      A,(BC)          ; FETCH ASCII DIGIT
025A  B7              :        OR      A               ; SET ZERO FLAG
025B  C8              :        RET     Z               ; RETURN IF FINISHED
025C  54              :        LD      D,H             ; SAVE CURRENT VALUE
025D  5D              :        LD      E,L             ; SAVE CURRENT VALUE
025E  29              :        ADD     HL,HL           ; TIMES TWO
025F  29              :        ADD     HL,HL           ; TIMES TWO
0260  19              :        ADD     HL,DE           ; ADD IN ORIGINAL VALUE
0261  29              :        ADD     HL,HL           ; TIMES TWO
0262  D630            :        SUB     48              ; ASCII BIAS
0264  FE0A            :        CP      10              ; CHECK FOR LEGAL VALUE
0266  3F              :        CCF
0267  D8              :        RET     C               ; RETURN IF ERROR
0268  5F              :        LD      E,A
0269  1600            :        LD      D,0
026B  19              :        ADD     HL,DE           ; ADD IN NEXT DIGIT
026C  03              :        INC     BC              ; INCREMENT POINTER
026D  18EA            :        JR      ADEC1-$
                      :*
                      :* THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                      :*  BY B,C AND CONVERTS THE ASCII HEXADECIMAL DIGITS INTO
                      :*  BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                      :*  SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                      :*
026F                  :AHEX    EQU     $
026F  21 0000         :        LD      HL,0            ; GET A 16-BIT ZERO
0272  0A              :AHEX1   LD      A,(BC)          ; FETCH ASCII DIGIT
0273  B7              :        OR      A               ; SET ZERO FLAG
0274  C8              :        RET     Z               ; RETURN IF DONE
0275  29              :        ADD     HL,HL           ; LEFT SHIFT
0276  29              :        ADD     HL,HL           ; LEFT SHIFT
```

PROGRAM ERRORS -- 0

```
0277  29              :       ADD     HL,HL           ; LEFT SHIFT
0278  29              :       ADD     HL,HL           ; LEFT SHIFT
0279  CD 96D0         :       CALL    AHS1            ; CONVERT TO BINARY
027C  FE10            :       CP      10H             ; CHECK FOR LEGAL VALUE
027E  3F              :       CCF
027F  D8              :       RET     C               ; RETURN IF ERROR
0280  85              :       ADD     A,L
0281  6F              :       LD      L,A
0282  03              :       INC     BC              ; INCREMENT POINTER
0283  18ED            :       JR      AHEX1-$
                      :*
                      :* THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
                      :*  ASCII HEXADECIMAL DIGITS AND STORES THEM IN MEMORY.
                      :*
0285                  :BINH   EQU     $
0285  47              :       LD      B,A             ; SAVE VALUE
0286  1F              :       RRA
0287  1F              :       RRA
0288  1F              :       RRA
0289  1F              :       RRA
028A  CD 93D0         :       CALL    BIN1
028D  77              :       LD      (HL),A
028E  23              :       INC     HL
028F  78              :       LD      A,B
0290  CD 93D0         :       CALL    BIN1            ; CONVERT TO ASCII
0293  77              :       LD      (HL),A
0294  C9              :       RET
                      :*
                      :* THIS ROUTINE CHECKS IF ANY PARAMETERS WERE ENTERED
                      :*  WITH THE COMMAND; IF NOT AN ERROR MESSAGE IS ISSUED.
                      :*
                      :*  CHECK FOR PRESENCE OF FILE NAME -- FATAL ERROR IF NOT
0295                  :CKFN   EQU     $
0295  CD 9F02         :       CALL    CKFN1           ; CHECK FOR FILE NAME
0298  C0              :       RET     NZ
0299  21 711B         :       LD      HL,MS1          ; 'MISSING FILE NAME'
029C  C3 6603         :       JP      MESS
                      :*  CHECK FOR PRESENCE OF FILE NAME -- NON-FATAL
029F                  :CKFN1  EQU     $
029F  2A E9F1         :       LD      HL,(FBUF)       ; CHECK FOR FILE NAME
02A2                  :CKFN2  EQU     $
02A2  7C              :       LD      A,H
02A3  B5              :       OR      L
02A4  C9              :       RET
                      :*  CHECK FOR PRESENCE OF 1ST ARG -- FATAL ERROR IF NOT
02A5                  :CKA1   EQU     $
02A5  CD AF02         :       CALL    CKA11           ; CHECK FOR VALID FIRST ARG
02A8  C0              :       RET     NZ
02A9  21 771B         :       LD      HL,MS2          ; 'MISSING ARGUMENT'
```

PROGRAM ERRORS -- 0

```
02AC  C3 6603           :       JP      MESS
                        :*  CHECK FOR PRESENCE OF 1ST ARG -- NON-FATAL
02AF                    :CKA11  EQU     $
02AF  2A FDF1           :       LD      HL,(ABUF)       ; CHECK FOR FIRST ARG
02B2  18EE              :       JR      CKFN2-$
                        :*  CHECK FOR PRESENCE OF 2ND ARG -- FATAL ERROR IF NOT
02B4                    :CKA2   EQU     $
02B4  CD BE02           :       CALL    CKA21           ; CHECK FOR VALID SECOND ARG
02B7  C0                :       RET     NZ
02B8  21 7E1B           :       LD      HL,MS3          ; 'MISSING SECOND ARG'
02BB  C3 6603           :       JP      MESS
                        :*  CHECK FOR PRESENCE OF 2ND ARG -- NON-FATAL
02BE                    :CKA21  EQU     $
02BE  2A 02F2           :       LD      HL,(ABUF+5)     ; CHECK FOR 2ND ARG
02C1  18DF              :       JR      CKFN2-$
                        :*
                        :* THIS ROUTINE CALLS THE FDOS DISK COMMUNICATION ROUTINE AND ISSUES
                        :*  AN ERROR MESSAGE IF AN ERROR IS DETECTED
                        :*
02C3                    :DCOM   EQU     $
02C3  CD 2220           :       CALL    DCOMM           ; FDOS DCOM
02C6  D0                :       RET     NC              ; CARRY SET MEANS ERROR
02C7  21 3E1C           :       LD      HL,MS40         ; DISK ERROR
02CA  C3 6603           :       JP      MESS
                        :*
                        :*  FECHK CHECKS FOR THE EXISTENCE OF THE PRIMARY FILE
                        :*    FATAL ERROR IF NO PRIMARY FILE
                        :*
02CD  3A 90F0           :FECHK  LD      A,(FILE0)       ; CHECK FOR A FILE NAME
02D0  B7                :       OR      A
02D1  C0                :       RET     NZ
02D2  C3 A409           :       JP      PFMS            ; ERROR -- NO FILE
                        :*
                        :* THIS ROUTINE IS USED TO FIND A LINE IN THE FILE AREA
                        :*  WHICH IS GREATER THAN OR EQUAL TO THE CURRENT LINE NO
                        :*
02D5                    :FIND   EQU     $
02D5  CD CD02           :       CALL    FECHK           ; PRIMARY FILE MUST EXIST
02D8  21 00F2           :       LD      HL,ABUF+3       ; BUFFER ADDRESS
02DB  22 E5F1           :       LD      (ADDS),HL       ; SAVE ADDRESS
02DE  2A 98F0           :FIND1  LD      HL,(BOFP)       ; BEGIN FILE ADDRESS
02E1  7C                :       LD      A,H             ; RETURN TO MONITOR IF
02E2  B5                :       OR      L               ;  FILE IS EMPTY ...
02E3  CA BA00           :       JP      Z,EOR
02E6  CD FE02           :FIND2  CALL    EOF1            ; CHECK FOR END OF FILE
02E9  ED5B E5F1         :       LD      DE,(ADDS)       ; FETCH FIND ADDR
02ED  3E04              :       LD      A,4
02EF  CD 11D1           :       CALL    ADR             ; BUMP LINE ADDRESS
02F2  CD 1503           :       CALL    COM0            ; COMPARE LINE NUMBERS

PROGRAM ERRORS --     0
```

```
02F5  D8                :         RET     C
02F6  C8                :         RET     Z
02F7  7E                :FIND3    LD      A,(HL)
02F8  CD 11D1            :         CALL    ADR
02FB  18E9               :         JR      FIND2-$
                         :*
                         :* WHEN SEARCHING THROUGH THE FILE AREA, THIS ROUTINE
                         :*  CHECKS TO SEE IF THE CURRENT ADDRESS IS THE END OF
                         :*  FILE
                         :*
02FD                     :EOF      EQU     $
02FD  23                 :         INC     HL
02FE                     :EOF1     EQU     $
02FE  3E01               :         LD      A,1             ; E-O-F INDICATOR
0300  BE                 :         CP      (HL)
0301  C0                 :         RET     NZ
0302  C3 BA00            :         JP      EOR
                         :*
                         :* THIS ROUTINE IS USED TO LOAD FOUR CHARACTERS FROM
                         :*  MEMORY INTO REGISTERS
                         :*
0305                     :LODM     EQU     $
0305  46                 :         LD      B,(HL)          ; FETCH CHARACTER
0306  23                 :         INC     HL
0307  4E                 :         LD      C,(HL)          ; FETCH CHARACTER
0308  23                 :         INC     HL
0309  56                 :         LD      D,(HL)          ; FETCH CHARACTER
030A  23                 :         INC     HL
030B  5E                 :         LD      E,(HL)          ; FETCH CHARACTER
030C  C9                 :         RET
                         :*
                         :* THIS ROUTINE STORES FOUR CHARACTERS FROM THE REGISTERS
                         :*  INTO MEMORY.
                         :*
030D                     :STOM     EQU     $
030D  73                 :         LD      (HL),E          ; STORE CHARACTER
030E  2B                 :         DEC     HL
030F  72                 :         LD      (HL),D          ; STORE CHARACTER
0310  2B                 :         DEC     HL
0311  71                 :         LD      (HL),C          ; STORE CHARACTER
0312  2B                 :         DEC     HL
0313  70                 :         LD      (HL),B          ; STORE CHARACTER
0314  C9                 :         RET
                         :*
                         :* THIS ROUTINE IS USED TO COMPARE TWO CHARACTER STRINGS
                         :*  OF LENGTH 4, ON RETURN ZERO FLAG SET MEANS BOTH
                         :*  STRINGS ARE EQUAL.  CARRY FLAG =0 MEANS STRING ADDRESSED
                         :*  BY D,E WAS GREATER THAN OR EQUAL TO CHARACTER STRING
                         :*  ADDRESSED BY H,L.
```

PROGRAM ERRORS -- 0

```
                        :*
                        :*
                        :* ON ENTRY TO THIS ROUTINE, D&E AND H&L POINT TO THE END
                        :*  OF THE STRINGS TO BE COMPARED
                        :*
0315                    :COM0    EQU     $
0315  0601               :        LD      B,1             ; EQUAL COUNTER
0317  0E04               :        LD      C,4             ; STRING LENGTH
0319  B7                 :        OR      A               ; CLEAR CARRY
031A  1A                 :COM01   LD      A,(DE)          ; FETCH CHARACTER
031B  9E                 :COM02   SBC     A,(HL)          ; COMPARE CHARACTERS
031C  2801               :        JR      Z,COM03-$
031E  04                 :        INC     B               ; INCREMENT EQUAL COUNTER
031F  1B                 :COM03   DEC     DE
0320  2B                 :        DEC     HL
0321  0D                 :        DEC     C
0322  20F6               :        JR      NZ,COM01-$
0324  05                 :        DEC     B
0325  C9                 :        RET
                        :*
                        :* THIS ROUTINE IS SIMILAR TO THE ABOVE ROUTINE EXCEPT ON
                        :*  RETURN CARRY FLAG = 0 MEANS THAT CHARACTER STRING
                        :*  ADDRESSED BY D,E IS ONLY > STRING ADDRESSED BY H,L.
                        :*
0326                    :COM1    EQU     $
0326  0E04               :        LD      C,4             ; STRING LENGTH
0328  1A                 :        LD      A,(DE)          ; FETCH CHARACTER
0329  D601               :        SUB     1
032B  18EE               :        JR      COM02-$
                        :*
                        :* THIS ROUTINE WILL TAKE ASCII CHARACTERS AND ADD ANY
                        :*  NECESSARY ASCII ZEROES SO THE RESULT IS A 4 CHARACTER
                        :*  ASCII VALUE.
                        :*
032D                    :NORM    EQU     $
032D  CD 0503            :        CALL    LODM            ; LOAD CHARACTERS
0330  AF                 :        XOR     A               ; FETCH A ZERO
0331  B8                 :        CP      B
0332  C8                 :        RET     Z
0333  BB                 :NORM1   CP      E
0334  C2 0D03            :        JP      NZ,STOM         ; STORE VALUES
0337  5A                 :        LD      E,D             ; NORMALIZE VALUE
0338  51                 :        LD      D,C
0339  48                 :        LD      C,B
033A  0630               :        LD      B,'0'
033C  18F5               :        JR      NORM1-$
                        :*
                        :*  PQ -- PRINT QUESTION; RETURN IF 'Y', POP AND RETURN
                        :*  IF 'N', POLL IF NOT 'Y' OR 'N'

PROGRAM ERRORS --       0
```

```
                        :*
033E                    :PQ       EQU     $
033E  CD 7800           :         CALL    PQ1             ; PRINT '?'
0341  CA BA00           :         JP      Z,EOR           ; RET W/ZERO SET MEANS 'N' TYPED
0344  C9                :         RET                     ; RETURN IF NOT 'N'
                        :*
                        :* PREPMS -- PRINT REPLACE MESSAGE AND WAIT FOR RESPONSE
                        :*
0345                    :PREPMS   EQU     $
0345  E5                :         PUSH    HL              ; SAVE HL
0346  21 FB1B           :         LD      HL,MS32         ; REPLACE MESSAGE
0349  CD 8A03           :         CALL    SCRN            ; PRINT IT
034C  E1                :         POP     HL              ; RESTORE HL
034D  C3 3E03           :         JP      PQ
                        :*
                        :* PDE ADR -- PRINT DE AS AN ADR; HL PTS TO D, NEXT LOC IS E
                        :*
0350                    :PDEADR   EQU     $
0350  56                :         LD      D,(HL)          ; GET D
0351  23                :         INC     HL
0352  5E                :         LD      E,(HL)          ; GET E
0353  CD 75D0           :PDEADR1  CALL    PDE             ; PRINT DE AS 4 HEX CHARS
0356  C3 42D0           :         JP      BLK1
                        :*
                        :* PED ADR -- PRINT ED AS AN ADR; HL PTS TO E, NEXT LOC IS D
                        :*
0359                    :PEDADR   EQU     $
0359  5E                :         LD      E,(HL)          ; GET E
035A  23                :         INC     HL
035B  56                :         LD      D,(HL)          ; GET D
035C  23                :         INC     HL
035D  18F4              :         JR      PDEADR1-$
                        :*
                        :*  CHLDE -- COMPARE H&L AGAINST D&E; CARRY=1 IF D&E>H&L
                        :*
035F                    :CHLDE    EQU     $
035F  7C                :         LD      A,H
0360  BA                :         CP      D
0361  D8                :         RET     C
0362  C0                :         RET     NZ
0363  7D                :         LD      A,L
0364  BB                :         CP      E
0365  C9                :         RET
                        :*
                        :* OUTPUT SYSTEM ERROR MESSAGE
                        :*
0366                    :MESS     EQU     $
0366  CD 6C03           :         CALL    MESS1
0369  C3 BA00           :         JP      EOR             ; RETURN TO EXECUTIVE

PROGRAM ERRORS --       0
```

```
036C                    :MESS1   EQU     $
036C  CD 3FD0           :        CALL    CRLF
036F  CD 5DD0           :        CALL    PCHAR          ; PRINT SYSTEM MSG INDICATOR
0372  24                :        DB      '$'
0373  CD 42D0           :        CALL    BLK1
0376  C3 51D0           :        JP      SCRNA
                        :*
                        :* THIS ROUTINE SCANS THROUGH A CHARACTER STRING UNTIL
                        :*  THE FIRST NON-BLANK CHARACTER IS FOUND. SBLK USES (PNTR)
                        :*  TO POINT TO STRING, SBLK1 USES HL
                        :* ON RETURN, CARRY SET INDICATES A CARRIAGE RETURN AS FIRST
                        :*  NON-BLANK CHARACTER.
                        :*
0379                    :SBLK    EQU     $
0379  2A 73F2           :        LD      HL,(PNTR)      ; FETCH ADDRESS
037C  7E                :SBLK1   LD      A,(HL)         ; FETCH CHARACTERS
037D  FE2C              :        CP      ','            ; SCAN OVER COMMA
037F  2803              :        JR      Z,SBLK2-$
0381  FE20              :        CP      ' '            ; CHECK FOR A BLANK
0383  C0                :        RET     NZ
0384  23                :SBLK2   INC     HL             ; POINT TO NEXT CHAR
0385  22 73F2           :        LD      (PNTR),HL      ; SAVE POINTER
0388  18F2              :        JR      SBLK1-$
                        :*
                        :* SCRN -- PRINT LINE PRECEEDED BY A <CR>
                        :*
038A                    :SCRN    EQU     $
038A  CD 3FD0           :        CALL    CRLF           ; <CR>
038D  C3 51D0           :        JP      SCRNA          ; PRINT MSG ENDING IN <CR>
                        :*
                        :*  GETSP -- GET SPCHAR AND MASK OUT HIGH ORDER BIT
                        :*
0390  3A F9F1           :GETSP   LD      A,(SPCHAR)     ; GET SPCHAR
0393  E67F              :        AND     7FH            ; MASK OUT HIGHEST BIT
0395  C9                :        RET
                        :*
                        :*  LINECNT -- DECREMENT THE COUNT OF THE NUMBER OF LINES PRINTED SO
                        :*    FAR, RETURN IF NOT ZERO, AND PROMPT THE USER TO CONTINUE PAGING
                        :*    IF EOL AND RESET LINE COUNT
                        :*
0396                    :LINECNT EQU     $
0396  3A F9F1           :        LD      A,(SPCHAR)     ; CHECK FOR PRINT
0399  FE50              :        CP      'P'
039B  C8                :        RET     Z              ; DO NOT PAGE IF PRINT
039C  B7                :        OR      A              ; SET FLAGS
039D  F8                :        RET     M
039E  3A 15F0           :        LD      A,(LPCNT)      ; GET LINE COUNT
03A1  3D                :        DEC     A
03A2  32 15F0           :        LD      (LPCNT),A      ; STORE NEW LINE COUNT

PROGRAM ERRORS --    0
```

```
03A5  C0              :         RET     NZ
03A6  CD 3E03         :         CALL    PQ
                      :*
                      :*  LINIT -- RESET LINE COUNT TO NUMBER OF LINES/PAGE
                      :*
03A9                  :LINIT    EQU     $
03A9  3A 65F2         :         LD      A,(NL)          ; RESET LINE COUNT
03AC  32 15F0         :         LD      (LPCNT),A
03AF  C9              :         RET

PROGRAM ERRORS --     0
```

```
                        :********                                        *.*******
                        :******** ARIAN COMMAND EXECUTION SECTION *.*******
                        :********                                        *.*******
```

PROGRAM ERRORS -- 0

```
                        ;**** COMMAND -- EXEC
                        ;*
                        ;* EXEC -- EXECUTE OBJECT CODE OR A TEXT FILE; THIS PROGRAM
                        ;*  CHECKS TO SEE IF A FILE NAME IS SPECIFIED IN THE PARAMETER
                        ;*  LIST, AND, IF SO, LOADS IT IF NECESSARY, MAKES THE FILE PRIMARY,
                        ;*  ASSEMBLES, AND EXECUTES IT; IF NO FILE NAME IS SPECIFIED, THE
                        ;*  OBJECT CODE STARTING AT THE ADDRESS SPECIFIED (OR IMPLIED BY THE
                        ;*  DEFAULT ASSEMBLY ADDRESS IF NO ADDRESS IS SPECIFIED) IS EXECUTED
                        ;*
03B0                    :EXEC   EQU     $
                        ;*  CHECK FOR BINARY FILE EXECUTE
03B0  FE42              :       CP      'B'
03B2  2848              :       JR      Z,EXECB-$
                        ;*  CHECK FOR FILE/NORMAL EXECUTE
03B4  CD 9F02           :       CALL    CKFN1           ; CHECK FOR FN
03B7  2819              :       JR      Z,EXECA-$       ; NO FN, SO NORMAL EXECUTE
                        ;*  EXECUTE THE SPECIFIED FILE
03B9  CD AF02           :       CALL    CKA11           ; NO NUMERIC ARGUMENT ALLOWED
03BC  203B              :       JR      NZ,PARAME-$     ; ERROR
                        ;*  CHECK TO SEE IF FILE IS LOCAL
03BE  CD AF0E           :       CALL    FSEA            ; SCAN LOCAL DIRECTORY FOR IT
03C1  2016              :       JR      NZ,EXEC1-$      ; NO LOAD IF FOUND
                        ;*  LOAD FILE FROM DISK IF NOT LOCAL
03C3  CD 4308           :       CALL    LOAD            ; LOAD FILE AND MAKE IT PRIMARY
                        ;*  CHECK FOR TYPE -- BINARY OR TEXT
03C6  3A 0FF0           :       LD      A,(LDTY)        ; CHECK FOR FILE TYPE
03C9  FE01              :       CP      1               ; BINARY FILE?
03CB  2818              :       JR      Z,EXECD-$       ; EXECUTE FROM ASMST IF BINARY
03CD  D2 6208           :       JP      NC,LOADE        ; ERROR IF NOT TYPE ZERO
03D0  180A              :       JR      EXEC2-$
                        ;*
                        ;*  -EXECA- EXECUTES EITHER CODE STARTING AT SPECIFIED
                        ;*    ADDRESS OR ASSEMBLES AND EXECUTES PRIMARY FILE
                        ;*
03D2  CD AF02           :EXECA  CALL    CKA11           ; CHECK FOR ARG
03D5  2014              :       JR      NZ,EXECR-$      ; EXECUTE IF ARG PRESENT
03D7  1803              :       JR      EXEC2-$
                        ;*
                        ;*  -EXEC1- MAKES THE SPECIFIED LOCAL FILE PRIMARY, ASSEMBLES
                        ;*    IT, AND EXECUTES IT
                        ;*
03D9  CD 5C0D           :EXEC1  CALL    FILE            ; MAKE FILE PRIMARY
                        ;*
                        ;*  -EXEC2- ASSEMBLES AND EXECUTES THE PRIMARY FILE
                        ;*
03DC  21 0000           :EXEC2  LD      HL,0            ; ZERO ABUF
03DF  22 FDF1           :       LD      (ABUF),HL
03E2  CD 5213           :       CALL    ASSM            ; ASSEMBLE PRIMARY FILE
                        ;*
```

PROGRAM ERRORS -- 0

```
                        :*  -EXECD- EXECUTES THE CODE POINTED TO BY THE DEFAULT
                        :*    EXECUTION ADDRESS, EXADR
                        :*
03E5  2A 0BF0           :EXECD  LD      HL,(EXADR)      ; SET UP EXECUTION ADDRESS
03E8  22 0DF2           :       LD      (BBUF),HL
                        :*
                        :*  -EXECR- EXECUTES THE CODE POINTED TO BY THE BINARY BUFFER,
                        :*    BBUF
                        :*
03EB  21 8A00           :EXECR  LD      HL,EOR          ; SET UP RETURN ADDRESS
03EE  31 00F3           :       LD      SP,STACK        ; SET UP STACK
03F1  E5                :       PUSH    HL              ; RETURN ADDRESS ON STACK
03F2  2A 0DF2           :       LD      HL,(BBUF)       ; EXECUTION ADDRESS
03F5  CD 3FD0           :       CALL    CRLF            ; NEW LINE
03F8  E9                :       JP      (HL)            ; RUN!!!
03F9  21 891B           :PARAME LD      HL,MS4          ; 'PARAMETER ERROR'
03FC  C3 6603           :       JP      MESS
                        :*
                        :*  -EXECB- EXECUTES EITHER THE SPECIFIED BINARY FILE OR THE
                        :*    CODE POINTED TO BY THE DEFAULT EXECUTION ADDRESS
                        :*
03FF  CD 9F02           :EXECB  CALL    CKFN1           ; MAY BE BINARY FILE NAME
0402  28E1              :       JR      Z,EXECD-$       ; EXECUTE CODE AT EXEC ADR
0404  CD 6A12           :       CALL    BFSCAN          ; CHECK FOR BINARY FILE
0407  CA 6808           :       JP      Z,DRSDF         ; NOT FOUND
040A  7E                :       LD      A,(HL)          ; <SP> ALSO MEANS NOT FOUND
040B  FE20              :       CP      ' '
040D  CA 6808           :       JP      Z,DRSDF
0410  11 0800           :       LD      DE,NMLEN        ; SKIP TO START ADDRESS OF BINARY FILE
0413  19                :       ADD     HL,DE           ; HL NOW POINTS TO LOW ORDER START
0414  5E                :       LD      E,(HL)          ; GET ADDRESS IN D&E
0415  23                :       INC     HL
0416  56                :       LD      D,(HL)
0417  EB                :       EX      DE,HL           ; NOW IN H&L
0418  22 0DF2           :       LD      (BBUF),HL       ; NEW EXECUTION ADDRESS
041B  18CE              :       JR      EXECR-$

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- TERM
                        :*
                        :*  LINK -- LINK WITH EXTERNAL SYSTEM
                        :*
041D                    :LINK   EQU     $
041D  CD 3FD0           :       CALL    CRLF            ; NEW LINE
0420  C3 00D0           :       JP      LINK1           ; JUMP TO LINK PROGRAM
```

PROGRAM ERRORS -- 0

```
                        :**** COMMAND -- CUST
                        :*
                        :*  THE CUST (CUSTOMIZE) COMMAND ALLOWS THE USER TO
                        :*  CREATE HIS OWN SET OF COMMANDS.  THE FORMAT OF THE
                        :*  COMMAND IS      CUST NAME ADR
                        :*
0423                    :CUST   EQU     $
0423  FE4C               :       CP      'L'
0425  CA 0B05            :       JP      Z,CUSTL
0428  FE53               :       CP      'S'             ; CHECK FOR CUSTS (SCRATCH) COMMAND
042A  CA 0605            :       JP      Z,CUSTS
042D  08                 :       EX      AF,AF'          ; SAVE A
042E  CD 9502            :       CALL    CKFN            ; CHECK FOR COMMAND NAME
0431  11 E9F1            :       LD      DE,FBUF         ; PTR TO NAME IN D&E
0434  08                 :       EX      AF,AF'          ; RESTORE A
0435  FE4E               :       CP      'N'             ; RENAME CUST CMND
0437  CA 9604            :       JP      Z,CUSTN
043A  FE44               :       CP      'D'             ; DELETE CUST CMND
043C  CA E604            :       JP      Z,CUSTD
043F  CD AF02            :       CALL    CKA11           ; CHECK FOR VALID ARG
0442  2006               :       JR      NZ,CUST0-$      ; PROCEED IF ARG PRESENT
                        :*  CREATE/REPLACE CUSTOMIZED COMMAND
0444  2A 0BF0            :       LD      HL,(EXADR)      ; GET DEFAULT EXECUTION ADR
0447  22 0DF2            :       LD      (BBUF),HL       ; STORE IN BBUF AS ADDRESS
044A  21 30F1            :CUST0  LD      HL,CUSTT
044D  46                 :       LD      B,(HL)          ; B=NO OF COMMANDS SO FAR
044E  04                 :       INC     B               ; CHECK FOR TOO MANY
044F  3E15               :       LD      A,NCUST+1
0451  B8                 :       CP      B
0452  CA 2605            :       JP      Z,CUSE1
0455  C5                 :       PUSH    BC              ; SAVE REGS
0456  E5                 :       PUSH    HL
0457  EB                 :       EX      DE,HL           ; D&E PT TO TABLE
0458  13                 :       INC     DE
0459  05                 :       DEC     B               ; B=NO CMNDS DEFINED SO FAR
045A  280F               :       JR      Z,CSTL2-$
045C  21 E9F1            :CUST1  LD      HL,FBUF
045F  0E04               :       LD      C,4             ; 4 BYTES PER CUST CMND
0461  CD 4201            :       CALL    SEAR            ; SCAN CUST TABLE FOR CMND
0464  CA 2C05            :       JP      Z,CSTL1
0467  13                 :       INC     DE              ; SKIP ADR
0468  13                 :       INC     DE
0469  10F1               :       DJNZ    CUST1-$
046B  E1                 :CSTL2  POP     HL              ; RESTORE REGS
046C  C1                 :       POP     BC
046D  34                 :       INC     (HL)            ; INCR NUMBER OF COMMANDS
046E  23                 :       INC     HL              ; POINT TO TABLE
046F  05                 :CUST2  DEC     B               ; 2ST ELT?
0470  2807               :       JR      Z,CUST3-$

PROGRAM ERRORS --     0
```

```
0472  3E06            :        LD      A,6             ; SKIP 6 BYTES
0474  CD 11D1          :        CALL    ADR
0477  18F6            :        JR      CUST2-$
0479  0E04            :CUST3   LD      C,4             ; STORE CMND (4 CHARS)
047B  11 E9F1          :        LD      DE,FBUF
047E  1A               :CUST4   LD      A,(DE)          ; GET COMMAND
047F  FE20             :        CP      ' '             ; PAD COMMAND W/BLANKS
0481  280E             :        JR      Z,CUST6-$
0483  77               :        LD      (HL),A          ; STORE COMMAND
0484  23               :        INC     HL
0485  13               :        INC     DE
0486  0D               :        DEC     C
0487  20F5             :        JR      NZ,CUST4-$
0489  ED5B 0DF2        :CUST5   LD      DE,(BBUF)       ; STORE ADR OF CMND
048D  73               :        LD      (HL),E
048E  23               :        INC     HL
048F  72               :        LD      (HL),D
0490  C9               :        RET
0491  CD DE12          :CUST6   CALL    CLERA           ; STORE BLANKS
0494  18F3             :        JR      CUST5-$
                       :*  RENAME CUSTOMIZED COMMAND
0496  CD CC04          :CUSTN   CALL    CTS             ; SEARCH CUSTOM TABLE
0499  1A               :        LD      A,(DE)          ; GET ADDRESS
049A  1B               :        DEC     DE
049B  67               :        LD      H,A             ; HIGH ORDER
049C  1A               :        LD      A,(DE)
049D  6F               :        LD      L,A             ; LOW ORDER
049E  22 0DF2          :        LD      (BBUF),HL       ; STORE ADDRESS IN BBUF
04A1  0E04             :        LD      C,4             ; BACK UP 4 BYTES
04A3  1B               :CSTN1   DEC     DE
04A4  0D               :        DEC     C
04A5  20FC             :        JR      NZ,CSTN1-$
04A7  EB               :        EX      DE,HL
04A8  22 E7F1          :        LD      (ADDS1),HL      ; STORE TEMPORRAILY
04AB  CD 3A05          :        CALL    PNN             ; 'NEW NAME?'
04AE  C8               :        RET     Z               ; ABORT IF JUST <CR>
04AF  11 E9F1          :        LD      DE,FBUF         ; STORE IN FBUF
04B2  0E04             :        LD      C,4             ; 4 CHARS
04B4  7E               :CSTN2   LD      A,(HL)          ; GET CHAR
04B5  FE0D             :        CP      0DH             ; <CR>?
04B7  280E             :        JR      Z,CSTN4-$
04B9  CD 05D1          :        CALL    CAPS            ; CONVERT LOWER CASE TO UPPER
04BC  12               :        LD      (DE),A          ; STORE CHAR
04BD  23               :        INC     HL              ; INCR PTRS
04BE  13               :        INC     DE
04BF  0D               :        DEC     C               ; DECR COUNT
04C0  20F2             :        JR      NZ,CSTN2-$
04C2  2A E7F1          :CSTN3   LD      HL,(ADDS1)      ; RESTORE POINTER TO TABLE ENTRY
04C5  18B2             :        JR      CUST3-$         ; HANDLE LIKE NEW CMND
```

PROGRAM ERRORS -- 0

```
04C7  3E20            :CSTN4   LD      A,' '
04C9  12              :        LD      (DE),A          ; STORE TRAILING <SP>
04CA  18F6            :        JR      CSTN3-$
                      :* CTS -- CUSTOMIZED TABLE SEARCH
04CC  21 30F1         :CTS     LD      HL,CUSTT        ; CUSTOM TABLE SEARCH FOR NAME IN FBUF
04CF  46              :        LD      B,(HL)          ; B=NO ELTS
04D0  78              :        LD      A,B
04D1  B7              :        OR      A               ; NONE?
04D2  CA 8905         :        JP      Z,CDERR1
04D5  23              :        INC     HL
04D6  3E04            :        LD      A,4
04D8  32 72F2         :        LD      (NCHR),A        ; 4 CHARS PER CMND
04DB  EB              :        EX      DE,HL
04DC  22 E5F1         :        LD      (ADDS),HL       ; STORE ADR
04DF  CD 2D01         :        CALL    COMS            ; SEARCH
04E2  C2 8905         :        JP      NZ,CDERR1       ; ERROR -- NOT FOUND
04E5  C9              :        RET
                      :* DELETE CUSTOMIZED COMMAND
04E6  CD CC04         :CUSTD   CALL    CTS             ; CUSTOM TABLE SEARCH
04E9  21 30F1         :        LD      HL,CUSTT
04EC  35              :        DEC     (HL)            ; DECR NO CMNDS
04ED  C8              :        RET     Z
04EE  23              :        INC     HL
04EF  D5              :        PUSH    DE
04F0  0E05            :        LD      C,5             ; BACK UP 5
04F2  1B              :CUSDL   DEC     DE              ; PT TO START OF CMND
04F3  0D              :        DEC     C               ; TO DELETE
04F4  20FC            :        JR      NZ,CUSDL-$
04F6  E1              :        POP     HL
04F7  23              :        INC     HL
04F8  0E06            :CUSML   LD      C,6             ; MOVE CMNDS
04FA  7E              :        LD      A,(HL)          ; CHECK FOR 0
04FB  B7              :        OR      A               ; DONE IF SO
04FC  C8              :        RET     Z
04FD  7E              :CUSM1   LD      A,(HL)          ; GET CHAR
04FE  12              :        LD      (DE),A          ; STORE CHAR
04FF  13              :        INC     DE
0500  23              :        INC     HL
0501  0D              :        DEC     C               ; DECR COUNT
0502  20F9            :        JR      NZ,CUSM1-$
0504  18F2            :        JR      CUSML-$
                      :* SCRATCH CUSTOMIZED COMMANDS
0506  AF              :CUSTS   XOR     A               ; SCRATCH ALL CUST COMMANDS
0507  32 30F1         :        LD      (CUSTT),A
050A  C9              :        RET
                      :* LIST CUSTOMIZED COMMANDS
050B  21 30F1         :CUSTL   LD      HL,CUSTT        ; DISPLAY CMNDS
050E  56              :        LD      D,(HL)          ; D=NO CMNDS
050F  7A              :        LD      A,D             ; NO CMNDS?
```

PROGRAM ERRORS -- 0

```
0510  B7                :         OR      A
0511  C8                :         RET     Z
0512  23                :         INC     HL              ; POINT TO FIRST
0513  0604              :CUSP1    LD      B,4             ; 4 CHARS
0515  CD 3FD0            :         CALL    CRLF
0518  7E                :CUSTP    LD      A,(HL)          ; GET CHAR
0519  CD 1ED0            :         CALL    OUTPUT          ; PRINT CHAR
051C  23                :         INC     HL
051D  10F9              :         DJNZ    CUSTP-$
051F  CD 620E            :         CALL    FPTRP           ; PRINT ADR
0522  15                :         DEC     D
0523  20EE              :         JR      NZ,CUSP1-$
0525  C9                :         RET
0526  21 281C            :CUSE1    LD      HL,MS38         ; 'TABLE OVERFLOW'
0529  C3 6603            :         JP      MESS
                         :*  REPLACE QUERY FOR CUSTOMIZED COMMAND REPLACEMENT
052C  CD 4503            :CSTL1    CALL    PREPMS          ; REPLACE QUERY
052F  EB                 :         EX      DE,HL
0530  ED5B 0DF2           :         LD      DE,(BBUF)       ; GET ADR OF CMND
0534  73                 :         LD      (HL),E
0535  23                 :         INC     HL
0536  72                 :         LD      (HL),D          ; STORE NEW ADR
0537  C3 BA00            :         JP      EOR             ; RETURN TO CONTROL
                         :*  PRINT 'NEW NAME?' AND CHECK FOR <CR> RESPONSE
053A  21 0E1C            :PNN      LD      HL,MS35         ; 'NEW NAME?'
053D  CD 8A03            :         CALL    SCRN
0540  CD 0101            :         CALL    READ            ; READ INPUT LINE
0543  7E                 :         LD      A,(HL)          ; CHECK FIRST CHAR FOR <CR>
0544  FE0D              :         CP      0DH
0546  C9                :         RET

PROGRAM ERRORS --      0
```

```
                        :****** COMMAND -- LSCR
                        :*
                        :*  SCR -- THIS ROUTINE CLEARS THE FILE DIRECTORY (MEMORY);
                        :*    IT CLEARS THE BINARY FILE DIRECTORY IF 'LSCRB' IS
                        :*    GIVEN AND THE TEXT FILE DIRECTORY IF JUST 'LSCR '
                        :*    IS SPECIFIED
                        :*
0547                    :SCR    EQU     $
0547  FE42              :       CP      'B'             ; BINARY FILE DIRECTORY SCRATCH
0549  CA E312           :       JP      Z,BSCR
054C  21 90F0           :SCR1   LD      HL,FILE0        ; SCRATCH ALL FILES
054F  0EA0              :       LD      C,MAXFIL*FELEN
0551  C3 FA00           :       JP      CLRZ            ; ZERO LOCAL DIRECTORY

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- EXIT
                        :*
                        :*  EXIT -- EXIT TO FDOS
                        :*
0554                    :EXIT   EQU     $
0554  CD 3FD0           :       CALL    CRLF            ; NEW LINE FOR FDOS PROMPT
0557  C3 2820           :       JP      FDOS

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- CMND
                        :*
                        :*  TURN OFF CL3
                        :*
055A  AF                :CMNDS   XOR     A               ; TURN OFF CL3; A=0
055B  32 DAF1           :        LD      (CDRIV),A
055E  C9                :        RET
                        :*
                        :*  TOGGLE CL3 DRIVE AND SET DRIVE NUMBER
                        :*
055F                    :CMND    EQU     $
055F  CD AF02           :        CALL    CKA11           ; CHECK FOR A FIRST ARGUMENT
0562  200B              :        JR      NZ,CDSET-$      ; SET COMMAND DRIVE NUMBER IF ARG PRESENT
0564  3A DAF1           :        LD      A,(CDRIV)       ; TOGGLE COMMAND DRIVE (AND FLAG)
0567  B7                :        OR      A               ; OFF?
0568  20F0              :        JR      NZ,CMNDS-$      ; TURN DRIVE OFF
056A  3C                :        INC     A               ; TURN ON DRIVE 1
056B  32 DAF1           :        LD      (CDRIV),A
056E  C9                :        RET
056F  3A 0DF2           :CDSET   LD      A,(BBUF)        ; GET DRIVE NUMBER
0572  B7                :        OR      A
0573  2808              :        JR      Z,CDERR-$       ; ERROR IF ZERO
0575  FE05              :        CP      5               ; ERROR IF > 4
0577  3004              :        JR      NC,CDERR-$
0579  32 DAF1           :        LD      (CDRIV),A       ; STORE AS COMMAND DRIVE NUMBER
057C  C9                :        RET
057D  21 E61B           :CDERR   LD      HL,MS27         ; INVALID CDRIVE
0580  C3 6603           :        JP      MESS
                        :*
                        :*  ROUTINE CMND1 SEARCHES FOR THE SPECIFIED COMMAND, LOADS IT
                        :*    IF FOUND, AND EXECUTES IT
                        :*
0583  3A DAF1           :CMND1   LD      A,(CDRIV)       ; CHECK FOR EXEC OPTION
0586  B7                :        OR      A
0587  2006              :        JR      NZ,CDCONT-$
0589  21 931B           :CDERR1  LD      HL,MS5          ; 'INVALID COMMAND'
058C  C3 6603           :        JP      MESS
058F  21 E9F1           :CDCONT  LD      HL,FBUF         ; SAVE ORIGINAL FBUF
0592  11 DBF1           :        LD      DE,SBUF
0595  01 0800           :        LD      BC,8
0598  EDB0              :        LDIR
059A  21 01FE           :        LD      HL,IBUF         ; STORE CMND NAME IN FBUF
059D  11 E9F1           :        LD      DE,FBUF
05A0  01 0400           :        LD      BC,4
05A3  EDB0              :        LDIR
05A5  21 911C           :        LD      HL,EXT          ; STORE '.CMD' EXTENSION
05A8  01 0400           :        LD      BC,4
05AB  EDB0              :        LDIR
                        :*  LOAD CL3 FILE

PROGRAM ERRORS --    0
```

```
05AD  CD B405             :        CALL    CLOAD           ; LOAD COMMAND FILE
                          :*  EXECUTE CL3 FILE
05B0  2A 0DF0             :        LD      HL,(CEXADR)     ; GET LOAD ADR
05B3  E9                  :        JP      (HL)            ; EXECUTE COMMAND
                          :*
                          :*  CLOAD SCANS THE DISK DIRECTORY FOR THE LEVEL 3 COMMAND
                          :*    NAME CONTAINED IN FBUF.  IF FOUND, IT LOADS THE FILE
                          :*    AND SETS UP THE EXECUTION ADDRESS.
                          :*
05B4  3A 0AF0             :CLOAD   LD      A,(DRIVEN)
05B7  32 D9F1             :        LD      (SDRIV),A       ; SAVE CURRENT DRIVE NUMBER
05BA  3A DAF1             :        LD      A,(CDRIV)       ; SET COMMAND DRIVE NUMBER
05BD  32 0AF0             :        LD      (DRIVEN),A
05C0  CD 250B             :        CALL    DSCAN           ; LOAD AND SCAN DIRECTORY FOR FILE NAME
05C3  3A D9F1             :        LD      A,(SDRIV)       ; RESTORE CURRENT DRIVE NUMBER
05C6  32 0AF0             :        LD      (DRIVEN),A
05C9  20BE                :        JR      NZ,CDERR1-$
05CB  E5                  :        PUSH    HL              ; SAVE PTR TO DIR ENTRY
05CC  21 DBF1             :        LD      HL,SBUF         ; RESTORE ORIGINAL FBUF
05CF  11 E9F1             :        LD      DE,FBUF
05D2  01 0800             :        LD      BC,8
05D5  EDB0                :        LDIR
05D7  E1                  :        POP     HL
05D8  3E08                :        LD      A,8             ; SKIP OVER FILE NAME
05DA  CD 11D1             :        CALL    ADR             ; HL PT TO DISK ADR PTR
05DD  5E                  :        LD      E,(HL)
05DE  23                  :        INC     HL
05DF  56                  :        LD      D,(HL)          ; DE PTS TO DISK ADR
05E0  23                  :        INC     HL
05E1  4E                  :        LD      C,(HL)          ; C=# BLOCKS
05E2  23                  :        INC     HL
05E3  23                  :        INC     HL
05E4  23                  :        INC     HL
05E5  D5                  :        PUSH    DE
05E6  5E                  :        LD      E,(HL)          ; DE=EXECUTION ADDRESS OF CL3 FILE
05E7  23                  :        INC     HL
05E8  56                  :        LD      D,(HL)
                          :*  SET UP EXECUTION ADDRESS OF CL3 FILE
05E9  ED53 0DF0           :        LD      (CEXADR),DE
05ED  E1                  :        POP     HL
05EE  3A DAF1             :        LD      A,(CDRIV)       ; DE PTS TO MEM, HL PTS TO DISK ADR
05F1  41                  :        LD      B,C             ; STORE NO BLOCKS IN B TEMP
05F2  4F                  :        LD      C,A             ; C=DRIVE NO
05F3  78                  :        LD      A,B             ; A=NO BLOCKS
05F4  0601                :        LD      B,1             ; READ
                          :*  LOAD CL3 FILE
05F6  C3 C302             :        JP      DCOM            ; GET FILE
```

```
                        :**** COMMAND -- RCVR
                        :*
                        :*  RCVR -- RECOVER FILE AND MAKE IT PRIMARY
                        :*
05F9                    :RCVR   EQU     $
05F9  CD A502           :       CALL    CKA1            ; CHECK FOR 1ST ARG
05FC  2A 0DF2           :       LD      HL,(BBUF)
05FF  7E                :       LD      A,(HL)          ; SAVE 1ST BYTE OF FILE
0600  E5                :       PUSH    HL
0601  F5                :       PUSH    AF
0602  CD 5C0D           :       CALL    FILE            ; CREATE FILE
0605  F1                :       POP     AF
0606  E1                :       POP     HL
0607  77                :       LD      (HL),A          ; RESTORE 1ST BYTE
0608  C3 D60B           :       JP      FFIX            ; FIX EOFP & MAXL

PROGRAM ERRORS --     0
```

```
                        ;**** COMMAND -- UTIL
                        ;*
                        ;*  MEMORY UTILITY SUBSYSTEM
                        ;*     THIS SUBSYSTEM GIVES THE USER A HOST OF MEMORY UTILITY
                        ;* COMMANDS WHICH ENABLE HIM TO EXAMINE MEMORY, DEPOSIT INTO
                        ;* MEMORY, FIND A BYTE STRING IN MEMORY, AND COPY A BLOCK
                        ;* OF MEMORY FROM ONE LOCATION TO ANOTHER.
                        ;*     THE UTILITY COMMANDS ARE
                        ;*        E (NNNN (NNNN)? )? -- EXAMINE MEMORY
                        ;*        D (NNNN)? -- DEPOSIT INTO MEMORY
                        ;*        S (NNNN)? -- SET DEFAULT POINTER
                        ;*        F NNNN NNNN (NN)+ -- FIND A BYTE STRING IN MEMORY
                        ;*        C NNNN NNNN NNNN -- COPY A BLOCK OF MEMORY
                        ;*        X -- EXIT TO ARIAN
                        ;*     THE USAGE OF THESE COMMANDS IS EXACTLY THE SAME AS THE
                        ;* CORRESPONDING COMMANDS IN MCS.
                        ;*
060B                    UTIL    EQU     $
060B  CD 3FD0           :       CALL    CRLF            ; PRINT PROMPT
060E  CD 5DD0           :       CALL    PCHAR
0611  2F                :       DB      '/'
0612  21 0B06           :       LD      HL,UTIL         ; SET UP RETURN ADDRESS
0615  E5                :       PUSH    HL
0616  CD 03D8           :       CALL    GETIN           ; GET COMMAND AND ARGUMENTS
0619  FE41               :       CP      'A'             ; IGNORE IF LESS THAN 'A'
061B  D8                :       RET     C
061C  FE45               :       CP      'E'             ; EXAMINE
061E  CA 18D8            :       JP      Z,UEXAM
0621  FE44               :       CP      'D'             ; DEPOSIT
0623  CA 1BD8            :       JP      Z,UDEPOS
0626  FE53               :       CP      'S'             ; SET/EXAMINE DEFAULT POINTER
0628  CA 24D8            :       JP      Z,USPTR
062B  FE46               :       CP      'F'             ; FIND
062D  CA 1ED8            :       JP      Z,UFIND
0630  FE43               :       CP      'C'             ; COPY
0632  CA 15D8            :       JP      Z,UCOPY
0635  FE58               :       CP      'X'             ; EXIT
0637  C2 3C06           :       JP      NZ,UERR         ; ERROR IF NOT
063A  E1                :       POP     HL              ; CLEAR STACK
063B  C9                :       RET                     ; RETURN TO ARIAN
063C  CD 57D0           :UERR   CALL    PEM             ; 'INVALID COMMAND' ERROR
063F  4943              :       DB      'IC'
0641  C9                :       RET                     ; RETURN TO UTIL

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- SETC
                        :*
                        :*  SETC -- SET/RESET REDIRECTABLE I/O DRIVERS;
                        :*    FORMS ARE  SETC = RESET I/O TO DEFAULT
                        :*               SETCI <ADR> = SET INPUT TO ROUTINE STARTING AT <ADR>
                        :*               SETCO <ADR> = SET OUTPUT TO ROUTINE STARTING AT <ADR>
                        :*
0642                    :SETC    EQU     $
0642  2A 0DF2           :        LD      HL,(BBUF)       ; GET ADR IN HL
0645  FE49              :        CP      'I'             ; INPUT?
0647  CA 0FD0           :        JP      Z,SETCI
064A  FE4F              :        CP      'O'             ; OUTPUT?
064C  CA 15D0           :        JP      Z,SETCO
064F  CD 12D0           :        CALL    RESCI           ; RESET I/O OTHERWISE TO DEFAULT
0652  C3 18D0           :        JP      RESCO

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- FIND
                        :*
                        :*  FINDS -- SEARCH THROUGH THE PRIMARY FILE AND FIND AND
                        :*        PRINT ALL OCCURRANCES OF THE SPECIFIED SEARCH STRING
                        :*  FORMS OF THIS COMMAND ARE   FIND(F,L,P) VERIFY? "<STRING>" <START LNUM>?
                        :*
 0655                   :FINDS   EQU     $
 0655  2A 0DF2          :        LD      HL,(BBUF)          ; GET START LINE NO
 0658  CD D502          :        CALL    FIND               ; H&L POINT TO LINE
 065B  23               :        INC     HL
 065C  11 13F2          :        LD      DE,S1BUF           ; CHECK FOR STRING
 065F  1A               :        LD      A,(DE)             ; GET '"' IF STRING PRESENT
 0660  FE22             :        CP      '"'                ; STRING PRESENT?
 0662  C2 F903          :        JP      NZ,PARAME
 0665  13               :        INC     DE                 ; DE PTS TO FIRST CHAR OF STRING
 0666  CD A903          :        CALL    LINIT              ; SET LINE COUNT
 0669  22 16F0          :FINDL   LD      (CLINE),HL         ; STORE CURRENT LINE PTR
 066C  01 0500          :        LD      BC,5               ; PT TO TEXT OF LINE
 066F  09               :        ADD     HL,BC
 0670  CD 9406          :        CALL    SSCAN              ; SCAN LINE
 0673  2819             :        JR      Z,EOLT-$           ; NOT FOUND
 0675  2A 16F0          :        LD      HL,(CLINE)
 0678  CD 7410          :        CALL    SCRNOC             ; PRINT LINE
 067B  23               :        INC     HL                 ; H&L PT TO NEXT LINE
 067C  CD 9603          :        CALL    LINECNT            ; CHECK PAGING
 067F  3A E9F1          :        LD      A,(FBUF)           ; CHECK FOR VERIFY OPTION
 0682  FE56             :        CP      'V'
 0684  2008             :        JR      NZ,EOLT-$
 0686  CD 1BD0          :        CALL    INPUT              ; GET CHAR
 0689  FE1B             :        CP      1BH                ; ABORT IF <ESC>
 068B  CA BA00          :        JP      Z,EOR
 068E  CD FE02          :EOLT    CALL    EOF1               ; EOF TEST
 0691  23               :        INC     HL                 ; PT TO FIRST CHAR OF NEXT LINE
 0692  18D5             :        JR      FINDL-$
                        :*  THIS ROUTINE SCANS THE INPUT LINE FOR THE STRING
 0694  1A               :SSCAN   LD      A,(DE)             ; SCAN LINE FOR STRING
 0695  BE               :        CP      (HL)
 0696  280B             :        JR      Z,SSCAN1-$
 0698  7E               :        LD      A,(HL)             ; GET CHAR
 0699  FE0D             :        CP      0DH                ; DONE?
 069B  2803             :        JR      Z,SSCAND-$
 069D  23               :        INC     HL
 069E  18F4             :        JR      SSCAN-$
 06A0  23               :SSCAND  INC     HL                 ; POINT TO NEXT LINE
 06A1  AF               :        XOR     A                  ; ZERO MEANS NOT FOUND
 06A2  C9               :        RET
                        :*  FIRST CHAR OF STR FOUND; SCAN FOR REST
 06A3  E5               :SSCAN1  PUSH    HL                 ; SCAN FOR REST OF STR
 06A4  D5               :        PUSH    DE

PROGRAM ERRORS --      0
```

```
06A5  23              :SSCAN2  INC     HL
06A6  13              :        INC     DE
06A7  1A              :        LD      A,(DE)          ; GET STR CHAR
06A8  FE0D            :        CP      0DH             ; END OF STR?
06AA  2808            :        JR      Z,SSCAN3-$
06AC  BE              :        CP      (HL)            ; COMPARE
06AD  28F6            :        JR      Z,SSCAN2-$
06AF  D1              :        POP     DE              ; CONTINUE SCAN
06B0  E1              :        POP     HL
06B1  23              :        INC     HL
06B2  18E0            :        JR      SSCAN-$
06B4  D1              :SSCAN3  POP     DE              ; CLEAR STACK
06B5  E1              :        POP     HL
06B6  B7              :        OR      A               ; NOT ZERO MEANS FOUND
06B7  C9              :        RET

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- EDIT
                        :*
                        :*  LEDIT -- THE INTRA-LINE EDITOR
                        :*        FORMAT   LEDIT <LINE NUMBER>
                        :*
                        :*  THE LEDIT COMMANDS ARE
                        :*        <SP>     COPY CHARACTER FROM OLD LINE TO NEW LINE AND
                        :*                 ADVANCE OLD LINE AND NEW LINE POINTERS
                        :*        E        SKIP TO END OF OLD LINE, COPYING CHARACTERS
                        :*                 DURING THE ADVANCE (LIKE MANY <SP>'S)
                        :*        D        DELETE CHARACTER POINTED TO BY OLD LINE POINTER
                        :*                 (DELETE NEXT CHAR); DELETED CHARACTERS ARE ENCLOSED
                        :*                 IN BACKSLASHES ('-')
                        :*        R        REPLACE CHARACTER(S) POINTED TO BY OLD LINE POINTER
                        :*                 WITH THE FOLLOWING STRING; BOTH POINTERS ARE ADVANCED;
                        :*                 NEW CHARACTERS ARE ECHOED ONTO THE I/O DEVICE;
                        :*                 REPLACEMENT IS ENDED WITH <ESC>
                        :*        I        INSERT A STRING OF CHARACTERS IN FRONT OF THE CHARACTER
                        :*                 CURRENTLY POINTED TO BY THE OLD LINE POINTER;
                        :*                 INSERTION IS ENDED WITH <ESC>; INSERTED CHARACTERS ARE
                        :*                 ENCLOSED IN SLASHES ('/'); ONLY NEW LINE POINTER IS
                        :*                 ADVANCED
                        :*        <BS>     BACK UP NEW LINE POINTER; A <BS> IS ECHOED AS THE
                        :*                 RESULT; PREVIOUS CHARACTERS ARE DELETED; ONLY THE
                        :*                 NEW LINE POINTER IS AFFECTED
                        :*        <DEL>    BACK UP NEW LINE POINTER; CHARACTERS BACKED OVER ARE
                        :*                 ENCLOSED IN '<' AND '>'; THESE CHARACTERS ARE DELETED;
                        :*                 ONLY NEW LINE POINTER IS AFFECTED
                        :*        <CR>     TERMINATE CREATION OF THE NEW LINE; THE CURRENT
                        :*                 NEW LINE IS SUBSTITUTED FOR THE OLD LINE;
                        :*                 IF <CR> IS THE FIRST EDITING CHAR, EDITING IS ABORTED
                        :*        S <LET>  SKIP TO SPECIFIED LETTER IN OLD LINE;
                        :*                 BOTH OLD AND NEW LINE POINTERS ARE ADVANCED, AND
                        :*                 THE CORRESPONDING CHARACTERS ARE PRINTED UNTIL
                        :*                 THE DESIRED CHARACTER IS ENCOUNTERED OR EOL;
                        :*                 WHEN FINISHED, THE OLD LINE POINTER POINTS TO THE
                        :*                 DESIRED CHARACTER (THIS CHARACTER IS NOT PRINTED);
                        :*                 THIS IS A TWO-CHARACTER COMMAND; NEITHER CHARACTER
                        :*                 WILL BE ECHOED; IT PERMITS INSERTION BEFORE THE
                        :*                 CHARACTER SPECIFIED, FOR EXAMPLE
                        :*        A        ABORT EDITING OF OLD LINE
                        :*        P        TERMINATE NEW LINE AT CURRENT NEW LINE POINTER,
                        :*                 MAKE THE NEW LINE THE NEW OLD LINE, AND EDIT
                        :*                 AND PRINT THE LINE
                        :*        X        EXIT AND REEDIT OLD LINE
                        :*
 06B8                   :LEDIT   EQU      $
 06B8  CD A502          :        CALL     CKA1            ; MUST HAVE <LNUM>
 06BB  CD D502          :        CALL     FIND            ; FIND IT (H&L PT TO IT)

PROGRAM ERRORS --      0
```

```
06BE  CD 3FD0          :LEDT0   CALL    CRLF
06C1  11 00F3          :        LD      DE,LOLD          ; SAVE LINE IN OLD LINE BUFFER
06C4  4E               :        LD      C,(HL)
06C5  0D               :        DEC     C                ; OMIT COUNT AT START
06C6  23               :        INC     HL
06C7  CD 42D0          :        CALL    BLK1             ; PRINT BLANK
06CA  7E               :SOLD    LD      A,(HL)           ; GET CHAR
06CB  CD 1ED0          :        CALL    OUTPUT           ; PRINT CHAR
06CE  12               :        LD      (DE),A           ; STORE CHAR
06CF  23               :        INC     HL               ; INCR PTRS
06D0  13               :        INC     DE
06D1  0D               :        DEC     C                ; DECR CHR COUNT
06D2  20F6             :        JR      NZ,SOLD-$
06D4  21 01FE          :LSTRT   LD      HL,IBUF          ; USE IBUF AS NEW LINE BUFFER
06D7  11 00F3          :        LD      DE,LOLD
06DA  CD 3FD0          :        CALL    CRLF
06DD  CD 5DD0          :        CALL    PCHAR            ; PRINT PROMPT
06E0  3F               :        DB      '?'
06E1  CD 21D0          :        CALL    IN8              ; CHECK FOR ABORTING <CR>
06E4  FE0D             :        CP      0DH
06E6  C8               :        RET     Z
06E7  1803             :        JR      LEDC0-$
                       :*
                       :*  GET COMMAND
                       :*          HL=NEW LINE PTR, DE=OLD LINE PTR
                       :*
06E9  CD 21D0          :LEDC    CALL    IN8              ; GET COMMAND
06EC  CD 05D1          :LEDC0   CALL    CAPS             ; CAPITALIZE
                       :*   <BS>=BACK UP
06EF  FE08             :        CP      8                ; <BS>
06F1  2005             :        JR      NZ,LEDC00-$
06F3  CD 3508          :        CALL    LBACK            ; BACK UP PTR IF POSSIBLE
06F6  18F1             :        JR      LEDC-$
                       :*   'A' = ABORT
06F8  FE41             :LEDC00  CP      'A'              ; ABORT?
06FA  2005             :        JR      NZ,LEDC1-$
06FC  0624             :        LD      B,'$'            ; ECHO '$'
06FE  C3 24D0          :        JP      OUT8
                       :*   ' ' = COPY
0701  FE20             :LEDC1   CP      ' '              ; <SP> IS COPY
0703  2029             :        JR      NZ,LEDC2-$
0705  CD 0B07          :        CALL    LCHR
0708  C3 E906          :        JP      LEDC
070B  1A               :LCHR    LD      A,(DE)           ; GET OLD CHAR
070C  FE0D             :        CP      0DH              ; EOL?
070E  280C             :        JR      Z,LEOL-$
0710  CD 1ED0          :LCHR1   CALL    OUTPUT           ; ECHO CHAR
0713  77               :        LD      (HL),A           ; STORE AS NEW
0714  13               :        INC     DE               ; INCR PTRS

PROGRAM ERRORS --      0
```

```
0715  23              :        INC     HL
0716  3E79            :        LD      A,IBUF+BUFLEN   ; EOL?
0718  BD              :        CP      L
0719  2807            :        JR      Z,LEOLE-$
071B  C9              :        RET
071C  CD 57D0         :LEOL    CALL    PEM             ; '****' MEANS EOL ENCOUNTERED
071F  2A2A            :        DB      '**'
0721  C9              :        RET
0722  E1              :LEOLE   POP     HL              ; GET RID OF RET ADDR
0723  21 7B1C         :        LD      HL,MS80
0726  CD 8A03         :        CALL    SCRN            ; PRINT REEDIT ERROR
0729  CD 3FD0         :LEDX    CALL    CRLF
072C  18A6            :        JR      LSTRT-$
                      :*  'X' = REEDIT (EXIT)
072E  FE58            :LEDC2   CP      'X'             ; REEDIT (EXIT)
0730  28F7            :        JR      Z,LEDX-$
                      :*  <CR> = TERMINATE
0732  FE0D            :        CP      0DH             ; DONE
0734  CA 2308         :        JP      Z,LEDD
                      :*  'S' = SKIP
0737  FE53            :        CP      'S'             ; SKIP
0739  2019            :        JR      NZ,LEDC4-$
073B  CD 21D0         :        CALL    IN8             ; CHAR TO SKIP TO
073E  48              :        LD      C,B             ; STORE IN C
073F  1A              :        LD      A,(DE)          ; GET 1ST CHAR
0740  FE0D            :        CP      0DH             ; EOL?
0742  28A5            :        JR      Z,LEDC-$
0744  CD 0B07         :        CALL    LCHR            ; STORE CHAR
0747  1A              :LEDC3   LD      A,(DE)          ; GET CHAR
0748  B9              :        CP      C               ; FOUND?
0749  289E            :        JR      Z,LEDC-$
074B  FE0D            :        CP      0DH             ; EOL?
074D  289A            :        JR      Z,LEDC-$
074F  CD 0B07         :        CALL    LCHR            ; STORE CHAR
0752  18F3            :        JR      LEDC3-$
                      :*  'E' = SKIP TO EOL
0754  FE45            :LEDC4   CP      'E'             ; SKIP TO EOL
0756  200A            :        JR      NZ,LEDC6-$
0758  1A              :LEDC5   LD      A,(DE)          ; CHECK FOR EOL
0759  FE0D            :        CP      0DH             ; <EOL>
075B  288C            :        JR      Z,LEDC-$
075D  CD 0B07         :        CALL    LCHR
0760  18F6            :        JR      LEDC5-$
                      :*  'D' = DELETE
0762  FE44            :LEDC6   CP      'D'             ; DELETE
0764  2025            :        JR      NZ,LEDC7-$
0766  CD 48D0         :        CALL    BSLSH           ; ENCLOSE IN '-'
0769  1A              :LDEL    LD      A,(DE)          ; GET CHAR TO DELETE
076A  FE0D            :        CP      0DH             ; EOL?
```

PROGRAM ERRORS -- 0

```
076C  2817            :        JR      Z,DEOL-$
076E  47              :        LD      B,A             ; ECHO IT
076F  CD 24D0         :        CALL    OUT8
0772  13              :        INC     DE
0773  CD 21D0         :        CALL    IN8
0776  CD 05D1         :        CALL    CAPS            ; CAPITALIZE
0779  FE44            :        CP      'D'             ; ANOTHER DELETE
077B  28EC            :        JR      Z,LDEL-$
077D  4F              :        LD      C,A
077E  CD 48D0         :        CALL    BSLSH           ; CLOSING '-'
0781  79              :        LD      A,C
0782  C3 EC06         :        JP      LEDC0
0785  CD 1C07         :DEOL    CALL    LEOL            ; PRINT '**' FOR EOL
0788  C3 E906         :        JP      LEDC
                      :*  'I' = INSERT
078B  FE49            :LEDC7   CP      'I'             ; INSERT
078D  2026            :        JR      NZ,LEDC8-$
078F  CD 5DD0         :        CALL    PCHAR           ; ENCLOSE IN '/'
0792  2F              :        DB      '/'
0793  CD 21D0         :LINS    CALL    IN8             ; GET CHAR
0796  FE1B            :        CP      1BH             ; <ESC> TO EXIT
0798  2814            :        JR      Z,LINS1-$
079A  FE0D            :        CP      0DH             ; END ABRUPTLY IF <CR>
079C  CA 1F08         :        JP      Z,LEDDB
079F  FE08            :        CP      8               ; <BS>
07A1  2005            :        JR      NZ,LINS0-$
07A3  CD 3508         :        CALL    LBACK           ; BACK UP POINTER IF POSSIBLE
07A6  18EB            :        JR      LINS-$
07A8  CD 1007         :LINS0   CALL    LCHR1           ; STORE AND ECHO CHAR
07AB  1B              :        DEC     DE              ; CORRECT FOR INSERTION
07AC  18E5            :        JR      LINS-$
07AE  CD 5DD0         :LINS1   CALL    PCHAR           ; CLOSING '/'
07B1  2F              :        DB      '/'
07B2  C3 E906         :        JP      LEDC
                      :*  'R' = REPLACE
07B5  FE52            :LEDC8   CP      'R'             ; REPLACE
07B7  2023            :        JR      NZ,LEDC9-$
07B9  CD 21D0         :LREP    CALL    IN8             ; GET CHAR
07BC  FE1B            :        CP      1BH             ; <ESC> TO STOP
07BE  CA E906         :        JP      Z,LEDC          ; GET NEXT COMMAND
07C1  FE0D            :        CP      0DH             ; <CR> TO FINISH EDIT
07C3  285E            :        JR      Z,LEDD-$
07C5  FE08            :        CP      8               ; <BS>
07C7  2005            :        JR      NZ,LREP0-$
07C9  CD 3508         :        CALL    LBACK           ; BACK UP POINTER IF POSSIBLE
07CC  18EB            :        JR      LREP-$
07CE  1A              :LREP0   LD      A,(DE)          ; CHECK FOR EOL
07CF  FE0D            :        CP      0DH
07D1  2003            :        JR      NZ,LREP1-$

PROGRAM ERRORS --     0
```

```
07D3  78              :        LD      A,B             ; GET NEW CHAR
07D4  18D2            :        JR      LINSO-$
07D6  78              :LREP1   LD      A,B             ; GET NEW CHAR
07D7  CD 1007         :        CALL    LCHR1           ; STORE CHAR
07DA  18DD            :        JR      LREP-$
                      :*  <DEL> = BACKUP
07DC  FE7F            :LEDC9   CP      7FH             ; <DEL> TO BACKUP
07DE  201E            :        JR      NZ,LPRIN-$
07E0  CD 5DD0         :        CALL    PCHAR           ; ENCL IN '<'
07E3  3C              :        DB      '<'
07E4  3E01            :LBAC    LD      A,IBUF          ; NOT POSSIBLE?
07E6  BD              :        CP      L
07E7  2830            :        JR      Z,LBACE-$
07E9  2B              :        DEC     HL              ; BACK UP
07EA  46              :        LD      B,(HL)          ; ECHO CHAR
07EB  CD 24D0         :        CALL    OUT8
07EE  CD 21D0         :        CALL    IN8             ; GET NEXT
07F1  FE7F            :        CP      7FH
07F3  28EF            :        JR      Z,LBAC-$
07F5  4F              :        LD      C,A             ; STORE IN C
07F6  CD 5DD0         :        CALL    PCHAR           ; END '>'
07F9  3E              :        DB      '>'
07FA  79              :        LD      A,C
07FB  C3 EC06         :        JP      LEDC0
                      :*  'P' = PRINT AND EDIT NEW LINE
07FE  FE50            :LPRIN   CP      'P'             ; PRINT AND EDIT NEW LINE
0800  200F            :        JR      NZ,LEDCE-$
0802  360D            :        LD      (HL),0DH        ; PUT IN ENDING <CR>
0804  7D              :        LD      A,L
0805  D601            :        SUB     IBUF
0807  C602            :        ADD     A,2             ; A=NO BYTES TO LINE
0809  4F              :        LD      C,A             ; C=NO BYTES
080A  21 00FE         :        LD      HL,IBUF-1       ; STORE
080D  71              :        LD      (HL),C
080E  C3 BE06         :        JP      LEDT0
                      :*  <BEL> = ERROR
0811  0607            :LEDCE   LD      B,'G'-40H       ; <BEL> MEANS UNRECOGNIZED
0813  CD 24D0         :        CALL    OUT8
0816  C3 E906         :        JP      LEDC
0819  CD 1C07         :LBACE   CALL    LEOL            ; BOL=EOL INDIC
081C  C3 E906         :        JP      LEDC
081F  CD 5DD0         :LEDDB   CALL    PCHAR           ; END INSERTION W/<CR>
0822  2F              :        DB      '/'
0823  360D            :LEDD    LD      (HL),0DH        ; WRITE <CR>, EOL
0825  23              :        INC     HL
0826  3601            :        LD      (HL),1
0828  0601            :        LD      B,IBUF          ; COMPUTE LINE LENGTH
082A  7D              :        LD      A,L
082B  90              :        SUB     B

PROGRAM ERRORS --      0
```

```
082C  C602              :        ADD     A,2             ; A=NO CHARS
082E  21 00FE           :        LD      HL,IBUF-1
0831  77                :        LD      (HL),A
0832  C3 760F           :        JP      LINE            ; PROCESS NEW LINE
0835  3E01              :LBACK   LD      A,IBUF          ; CHECK FOR BEGINNING OF LINE
0837  BD                :        CP      L
0838  2804              :        JR      Z,LBACK1-$      ; ERROR IF SO
083A  2B                :        DEC     HL              ; BACK UP NEW LINE POINTER IF NOT
083B  C3 24D0           :        JP      OUT8            ; PRINT CHAR IN B (<BS>)
083E  0624              :LBACK1  LD      B,'$'           ; ECHO AS <ESC>
0840  C3 24D0           :        JP      OUT8

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- LOAD
                        :*
                        :*  LOAD -- LOAD DISK FILE
                        :*       FORMAT   LOAD <FILENAME> <ADR>
                        :*
                        :*  THE LOAD COMMAND LOADS THE DISK FILE TO THE
                        :*       ADDRESS SPECIFIED.  IF THE DISK FILE IS A TEXT FILE
                        :*       (TYPE 0), THE SPECIFIED ADDRESS IS USED; IF THE
                        :*       DISK FILE IS A BINARY FILE (TYPE 1), THE ADDRESS IN
                        :*       THE DISK DIRECTORY IS USED (GIVEN ADDRESS IS OVERRIDDEN).
                        :*  THE SPECIFIED FILE MUST NOT ALREADY EXIST IN THE
                        :*       LOCAL FILE DIRECTORY.  IF SO, AN ERROR WILL BE FLAGGED
                        :*       AND THE LOAD ABORTED.  ALSO, THE SAME IS TRUE FOR
                        :*       THE FILE NAME IN THE DISK DIRECTORY (IT MUST BE PRESENT).
                        :*  WHEN COMPLETED, THE LOCAL FILE DIRECTORY WILL HAVE THE
                        :*       NEW FILE AS ITS PRIMARY FILE.  IF IT IS A BINARY FILE,
                        :*       THE DIRECTORY WILL REFLECT THE NUMBER OF BLOCKS LOADED;
                        :*       IF IT IS A TEXT FILE, THE DIRECTORY WILL REFLECT THE
                        :*       ADDRESS RANGE OF THE TEXT.
                        :*
0843                    :LOAD    EQU     $
0843  CD 9502           :        CALL    CKFN            ; CHECK FOR FILE NAME
0846  2A 0DF2           :        LD      HL,(BBUF)       ; GET LOAD ADDRESS
0849  22 6AF2           :        LD      (ASMST),HL
084C  CD 5C0D           :        CALL    FILE            ; CREATE AND VERIFY FILE
084F  CD 250B           :        CALL    DSCAN           ; SCAN DIRECTORY (DISK)
0852  2014              :        JR      NZ,DRSDF-$      ; Z=0 MEANS FILE NOT FOUND
0854  E5               :        PUSH    HL              ; SAVE ADR OF FILE NAME
0855  3E0C             :        LD      A,FTDSP         ; GET DISPLACEMENT FOR TYPE
0857  CD 11D1           :        CALL    ADR             ; INCR H&L BY IT
085A  7E               :        LD      A,(HL)          ; GET FILE TYPE
085B  B7               :        OR      A               ; TYPE ZERO?
085C  2831             :        JR      Z,LDTF-$
085E  FE01             :        CP      1               ; TYPE 1?
0860  280C             :        JR      Z,LDBF-$
0862  21 C21B          :LOADE   LD      HL,MS22         ; ERROR -- TYPE
0865  C3 6603          :        JP      MESS
0868  21 191C          :DRSDF   LD      HL,MS36         ; NOT FOUND
086B  C3 6603          :        JP      MESS
086E  E1               :LDBF    POP     HL              ; GET ADR OF FILE (LOAD BINARY)
086F  E5               :        PUSH    HL              ; SAVE
0870  ED5B 6AF2        :        LD      DE,(ASMST)      ; GET LOAD ADDRESS
0874  7A               :        LD      A,D             ; CHECK TO SEE IF THERE IS ONE (NOT ZERO)
0875  B3               :        OR      E
0876  2010             :        JR      NZ,LDBF0-$
0878  3E0D             :        LD      A,FADSP         ; GET DISP FOR FILE ADR
087A  CD 11D1           :        CALL    ADR
087D  5E               :        LD      E,(HL)          ; D&E=ADR IN MEMORY
087E  23               :        INC     HL

PROGRAM ERRORS --     0
```

```
087F  56              :        LD      D,(HL)
0880  ED53 6AF2       :        LD      (ASMST),DE      ; STORE START ADDRESS OF BINARY FILE
0884  ED53 08F0       :        LD      (EXADR),DE      ; NEW EXEC ADR
0888  3E01            :LDBF0   LD      A,1             ; SET FOR BINARY LOAD
088A  32 0FF0         :        LD      (LDTY),A        ; LOAD TYPE
088D  1808            :        JR      LOADF-$
088F  2A 98F0         :LDTF    LD      HL,(BOFP)
0892  EB              :        EX      DE,HL           ; D&E PT TO ADR
0893  AF              :        XOR     A               ; SET FILE TYPE TO 0 (TEXT)
0894  32 0FF0         :        LD      (LDTY),A        ; STORE
0897  E1              :LOADF   POP     HL              ; GET ADR OF FILE IN DIR
0898  D5              :        PUSH    DE              ; SAVE ADR
0899  E5              :        PUSH    HL              ; SAVE ADR OF FILE
089A  3E0A            :        LD      A,LBDSP         ; LOAD BLOCK NO DISP
089C  CD 11D1         :        CALL    ADR
089F  4E              :        LD      C,(HL)          ; C=NO BLOCKS
08A0  E1              :        POP     HL              ; GET ADR
08A1  3E08            :        LD      A,LDSP          ; LOAD DISP
08A3  CD 11D1         :        CALL    ADR
08A6  7E              :        LD      A,(HL)          ; GET LOW ORDER
08A7  23              :        INC     HL
08A8  66              :        LD      H,(HL)          ; GET HIGH
08A9  6F              :        LD      L,A             ; LOW
08AA  D1              :        POP     DE              ; H&L PT TO DISK ADR, D&E PT TO MEM ADR, C=NO BL
08AB  0601            :        LD      B,1             ; READ FROM DISK
08AD  79              :        LD      A,C             ; A=NO BLOCKS
08AE  CD DC08         :        CALL    DRIVE           ; C=DRIVE NO
08B1  F5              :        PUSH    AF              ; SAVE A=NO BLOCKS
08B2  D5              :        PUSH    DE              ; SAVE MEMORY ADR
08B3  CD C302         :        CALL    DCOM            ; LOAD FILE
08B6  E1              :        POP     HL              ; GET MEMORY ADR
08B7  F1              :        POP     AF              ; GET A=NO BLOCKS
08B8  47              :        LD      B,A             ; SAVE IN B
08B9  3A 0FF0         :        LD      A,(LDTY)        ; GET LOAD TYPE
08BC  B7              :        OR      A               ; TYPE 0? (TEXT)
08BD  CA D60B         :        JP      Z,FFIX          ; FIX IF SO
08C0  78              :        LD      A,B             ; A=NO BLOCKS
08C1  84              :        ADD     A,H             ; COMPUTE EOFP (= NO BLOCKS + START ADR)
08C2  67              :        LD      H,A             ; STORE IN ASM END
08C3  2B              :        DEC     HL              ; ADJUST TO CORRECT VALUE
08C4  22 6CF2         :        LD      (ASMEN),HL
08C7  CD EE0C         :        CALL    FDEL            ; DELETE TEXT FILE CREATED
08CA  C3 B812         :        JP      BFDENT          ; CREATE BINARY FILE
                      :*
                      :*  DRIVEC CHECKS THE SPECIAL CHARACTER FOR A VALID DRIVE
                      :*    NUMBER (1-3) AND SETS THE DRIVE NUMBER TO THAT VALUE
                      :*    IF PRESENT
                      :*
08CD  3A F9F1         :DRIVEC  LD      A,(SPCHAR)      ; CHECK SPECIAL CHAR

PROGRAM ERRORS --     0
```

```
                        COMMAND -- LOAD

08D0  FE31              :DRIVEC1 CP      '1'             ; '1'?
08D2  D8                :        RET     C
08D3  FE35              :        CP      '5'             ; '5'?
08D5  D0                :        RET     NC
08D6  D630              :        SUB     '0'             ; ADJUST TO 1-3
08D8  32 0AF0           :        LD      (DRIVEN),A      ; SET DRIVE NUMBER
08DB  C9                :        RET
                        :*
                        :*  DRIVE LOADS C WITH THE DRIVE NUMBER AND AFFECTS NO OTHER
                        :*    REGISTERS
                        :*
08DC  08                :DRIVE   EX      AF,AF'          ; SAVE AF
08DD  3A 0AF0           :        LD      A,(DRIVEN)      ; GET DRIVE NUMBER
08E0  4F                :        LD      C,A             ; IN C
08E1  08                :        EX      AF,AF'          ; RESTORE AF
08E2  C9                :        RET
                        :
                        :
                        :*
                        :*  TYPE COMMAND -- REDEFINE A FILE'S TYPE
                        :*    COMMAND IS TYPE0 <FNAME> FOR RETYPING <FNAME> TO 0
                        :*    TYPE <FNAME> <HADR> TYPES <FNAME> AS 1
                        :*
08E3  CD 9502           :TYPE    CALL    CKFN            ; MUST HAVE FILE NAME
08E6  CD 2508           :        CALL    DSCAN           ; FIND FILE
08E9  C2 6808           :        JP      NZ,DRSDF        ; NOT FOUND
08EC  3E0C              :        LD      A,FTDSP         ; CHECK TYPE
08EE  CD 11D1           :        CALL    ADR
08F1  3A FAF1           :        LD      A,(SPCHR2)      ; GET SPECIFIED FILE TYPE
08F4  FE30              :        CP      '0'             ; ONLY 0 ALLOWED
08F6  280D              :        JR      Z,TYPEZ-$
08F8  3601              :        LD      (HL),1          ; IF NOT 0, TYPE 1
08FA  23                :        INC     HL              ; POINT TO ADDRESS
08FB  ED5B 0DF2         :        LD      DE,(BBUF)       ; GET EXEC ADDRESS (0 DEFAULT)
08FF  73                :        LD      (HL),E          ; STORE ADDRESS
0900  23                :        INC     HL
0901  72                :        LD      (HL),D
0902  C3 0808           :        JP      RPDIR           ; REPLACE DIRECTORY
0905  3600              :TYPEZ   LD      (HL),0          ; STORE TYPE ZERO
0907  C3 0808           :        JP      RPDIR

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- FCHK
                        :*
                        :*  FCHK -- FILE CHECK; CHECK TO SEE THAT THE SPECIFIED FILE
                        :*        IS VALID
                        :*
090A                    :FCHK   EQU     $
090A  CD CD02           :       CALL    FECHK           ; PRIMARY FILE MUST EXIST
090D  CD 9F02           :       CALL    CKFN1           ; CHECK FOR A FILE NAME
0910  200A               :       JR      NZ,FCHK0-$
0912  2A 9AF0            :       LD      HL,(EOFP)       ; SET UP EOF
0915  4D                :       LD      C,L             ; B&C=EOF
0916  44                :       LD      B,H
0917  2A 98F0            :       LD      HL,(BOFP)       ; H&L=BOF
091A  1812              :       JR      FCHK1-$
091C  CD AF0E           :FCHK0  CALL    FSEA            ; SEARCH FOR SPECIFIED FILE
091F  CA 6808           :       JP      Z,DRSDF
0922  11 0800           :       LD      DE,NMLEN        ; GET BOF
0925  19                :       ADD     HL,DE
0926  5E                :       LD      E,(HL)          ; D&E=BOF
0927  23                :       INC     HL
0928  56                :       LD      D,(HL)
0929  23                :       INC     HL
092A  4E                :       LD      C,(HL)          ; B&C=EOF
092B  23                :       INC     HL
092C  46                :       LD      B,(HL)
092D  EB                :       EX      DE,HL
092E  CD 3F09           :FCHK1  CALL    FCK             ; H&L=BOF
0931  7D                :       LD      A,L             ; CHECK EOF FOUND AGAINST STATED EOF
0932  B9                :       CP      C
0933  201D              :       JR      NZ,FCHKE-$
0935  7C                :       LD      A,H
0936  B8                :       CP      B
0937  2019              :       JR      NZ,FCHKE-$
0939  21 D01B           :       LD      HL,MS25         ; 'VALID FILE'
093C  C3 6603           :       JP      MESS
093F  7E                :FCK    LD      A,(HL)          ; A=NO BYTES IN LINE
0940  FE01              :       CP      1               ; EOF?
0942  C8                :       RET     Z
0943  57                :       LD      D,A             ; COUNT OF LAST LINE IN D
0944  85                :       ADD     A,L             ; COMPUTE EOL
0945  6F                :       LD      L,A
0946  3001              :       JR      NC,FCK1-$
0948  24                :       INC     H
0949  2B                :FCK1   DEC     HL              ; POINT TO LAST CHAR IN LINE
094A  7E                :       LD      A,(HL)          ; CHECK FOR ENDING <CR>
094B  FE0D              :       CP      0DH
094D  2003              :       JR      NZ,FCHKE-$      ; ERROR IF NOT <CR>
094F  23                :       INC     HL              ; FIRST CHAR OF NEW LINE
0950  18ED              :       JR      FCK-$

PROGRAM ERRORS --    0
```

```
0952  21 DB1B           :FCHKE   LD      HL,MS26         ; 'INVALID FILE'
0955  C3 6603           :        JP      MESS

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- DDEL
                        :*
                        :*  DDEL -- DELETE DISK FILE.  DELETE THE SPECIFIED DISK FILE
                        :*      FROM THE DIRECTORY.  FORMAT   DDEL DISKFILENAME
                        :*
0958                    :DELDF   EQU     $
0958  CD 9502           :        CALL    CKFN            ; DELETE DISK FILE (CHECK FOR FN)
0958  CD 2508           :        CALL    DSCAN           ; SCAN FOR FILE NAME
095E  C2 6808           :        JP      NZ,DRSDF
0961  0610               :        LD      B,16            ; 16 <SP>'S
0963  3620               :DLDF2   LD      (HL),' '        ; FILL WITH <SP>
0965  23                 :        INC     HL
0966  10FB               :        DJNZ    DLDF2-$
0968  C3 080B           :        JP      RPDIR           ; REPLACE DIR

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- DNAME
                        :*
                        :*  DNAME -- RENAME A DISK FILE
                        :*
0968                    :DNAME   EQU     $
0968  CD 9502           :        CALL    CKFN            ; CHECK FOR FILE NAME
096E  CD 2508           :        CALL    DSCAN
0971  C2 6808           :        JP      NZ,DRSDF        ; NOT FOUND
0974  CD 280C           :        CALL    RNMEE           ; FILE RENAME ENTRY POINT
0977  C3 0808           :        JP      RPDIR           ; REPLACE DIRECTORY

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- SAVE
                        :*
                        :*  SAVE -- THE SAVE COMMAND.  THIS COMMAND SAVES THE PRIMARY
                        :*      LOCAL FILE ON DISK UNDER THE NAME SPECIFIED IN THE
                        :*      COMMAND.  FORMAT   SAVE DISKFILENAME
                        :*     SAVE ALSO SAVES BINARY FILES; FORMAT IS
                        :*         SAVEB DISKFILENAME <STARTADR> <ENDADR>
                        :*      AND CHANGES THE TYPE OF AN EXISTING DISK FILE; FORMAT IS
                        :*         SAVET DISKFILENAME <HADR>?
                        :*
 097A                   :DSAVE   EQU     $
 097A  F5               :        PUSH    AF              ; SAVE SPCHAR
 097B  CD CD02          :        CALL    FECHK           ; PRIMARY FILE MUST EXIST
 097E  CD 9502          :        CALL    CKFN            ; CHECK FOR FILE NAME
 0981  F1               :        POP     AF              ; GET SPCHAR
 0982  FE54             :        CP      'T'             ; SAVE TYPE FOR SPECIFIED FILE
 0984  CA E308          :        JP      Z,TYPE
 0987  FE42             :        CP      'B'             ; BINARY SAVE
 0989  281F             :        JR      Z,SAVEB-$       ; SAVE BINARY FILE IF SO
 098B  2A 98F0          :        LD      HL,(BOFP)       ; GET ADR
 098E  22 12F0          :        LD      (SSTRT),HL      ; SAVE START (MEMORY) ADR
 0991  EB               :        EX      DE,HL
 0992  2A 9AF0          :        LD      HL,(EOFP)       ; GET END TO COMPUTE RANGE
 0995  7D               :        LD      A,L             ; COMPUTE NO BYTES TO SAVE
 0996  93               :        SUB     E
 0997  6F               :        LD      L,A
 0998  7C               :        LD      A,H
 0999  9A               :        SBC     A,D
 099A  67               :        LD      H,A
 099B  23               :        INC     HL              ; INCLUSIVE, SO ADD 1
 099C  22 10F0          :        LD      (SRANG),HL      ; SAVE RANGE
 099F  2B               :        DEC     HL              ; DECR FOR TEST
 09A0  7C               :        LD      A,H             ; NO RANGE? ... ERROR
 09A1  B5               :        OR      L
 09A2  2060             :        JR      NZ,SAV1-$
 09A4  21 F31B          :PFMS    LD      HL,MS31         ; 'NO PRIMARY FILE'
 09A7  C3 6603          :        JP      MESS
 09AA  3A FAF1          :SAVEB   LD      A,(SPCHAR+1)    ; CHECK 6TH CHAR
 09AD  CD D008          :        CALL    DRIVEC1         ; SPECIAL ENTRY POINT TO CHECK FOR DRIVE NUMBER
 09B0  CD AF02          :        CALL    CKA11           ; CHECK FOR ANY FIRST ARG
 09B3  200E             :        JR      NZ,SAVEB1-$     ; CHECK FOR SECOND ARG IF SO
 09B5  2A 6AF2          :        LD      HL,(ASMST)      ; GET RANGE FOR LAST ASSEMBLY
 09B8  22 0DF2          :        LD      (BBUF),HL       ; FOR RANGE FOR SAVE
 09BB  2A 6CF2          :        LD      HL,(ASMEN)
 09BE  22 0FF2          :        LD      (BBUF+2),HL
 09C1  1803             :        JR      SAVEB2-$
 09C3  CD B402          :SAVEB1  CALL    CKA2            ; CHECK FOR 2ND ARG
 09C6  2A 0DF2          :SAVEB2  LD      HL,(BBUF)       ; GET STARTING MEM ADR
 09C9  22 12F0          :        LD      (SSTRT),HL      ; SAVE AS START
```

```
09CC  EB              :        EX      DE,HL
09CD  2A 0FF2         :        LD      HL,(BBUF+2)     ; GET 2ND PARAM
09D0  7D              :        LD      A,L             ; COMPUTE RANGE
09D1  93              :        SUB     E
09D2  6F              :        LD      L,A
09D3  7C              :        LD      A,H
09D4  9A              :        SBC     A,D
09D5  67              :        LD      H,A
09D6  23              :        INC     HL              ; INCLUSIVE
09D7  22 10F0         :        LD      (SRANG),HL      ; SAVE RANGE
09DA  2B              :        DEC     HL              ; ERROR IF NO RANGE
09DB  7C              :        LD      A,H
09DC  B5              :        OR      L
09DD  285A            :        JR      Z,RNERR-$
09DF  CD 040A         :        CALL    SAV1            ; DO SAVE
09E2  3A 06F2         :        LD      A,(ABUF+9)      ; CHECK FOR THIRD ARG
09E5  B7              :        OR      A
09E6  2806            :        JR      Z,SAVEB3-$
09E8  2A 11F2         :        LD      HL,(BBUF+4)     ; GET THIRD ARG
09EB  22 0DF2         :        LD      (BBUF),HL       ; SAVE THIRD ARG AS EXEC ADR
09EE  CD 2508         :SAVEB3  CALL    DSCAN           ; SCAN FOR FILE NAME
09F1  3E0C            :        LD      A,FTDSP         ; SET TYPE TO 1
09F3  CD 11D1         :        CALL    ADR             ; POINT TO TYPE ENTRY IN DIR
09F6  3601            :        LD      (HL),1
09F8  23              :        INC     HL              ; POINT TO ADR
09F9  EB              :        EX      DE,HL
09FA  2A 0DF2         :        LD      HL,(BBUF)       ; GET START ADR
09FD  EB              :        EX      DE,HL
09FE  73              :        LD      (HL),E          ; SAVE START ADR
09FF  23              :        INC     HL
0A00  72              :        LD      (HL),D
0A01  C3 0808         :        JP      RPDIR           ; REPLACE NEW DIR
0A04  CD 560C         :SAV1    CALL    DIR             ; GET DIRECTORY
0A07  CD 2508         :        CALL    DSCAN           ; SCAN FOR FILE
0A0A  203E            :        JR      NZ,SAV2-$       ; CREATE FILE IF NOT FOUND
0A0C  CD 4503         :        CALL    PREPMS          ; REPLACE QUERY
0A0F  22 D5F1         :        LD      (SPSAV),HL      ; SAVE H&L
0A12  3E0A            :        LD      A,LBDSP         ; GET NO BLOCKS
0A14  CD 11D1         :        CALL    ADR
0A17  7E              :        LD      A,(HL)          ; A=NO BLOCKS
0A18  2A 10F0         :        LD      HL,(SRANG)      ; COMPARE AGAINST RANGE
0A1B  BC              :        CP      H
0A1C  3821            :        JR      C,SAVD-$        ; DELETE ENTRY IF TOO SMALL
0A1E  2004            :        JR      NZ,SAV1A-$
0A20  7D              :        LD      A,L             ; MATCH -- SO SEE IF EXACTLY 256 BYTES
0A21  B7              :        OR      A
0A22  201B            :        JR      NZ,SAVD-$
0A24  EB              :SAV1A   EX      DE,HL           ; PUT NO OF BLOCKS IN DIR ENTRY
0A25  2A D5F1         :        LD      HL,(SPSAV)      ; POINT TO DIR ENTRY
```

PROGRAM ERRORS -- 0

```
0A28  3E0A            :        LD      A,LBDSP         ; POINT TO BLOCK SUBENTRY
0A2A  CD 11D1         :        CALL    ADR
0A2D  7B              :        LD      A,E             ; CHECK FOR EXACT COUNT
0A2E  B7              :        OR      A
0A2F  2801            :        JR      Z,SAV1B-$
0A31  14              :        INC     D               ; MAKE BLOCK COUNT INCLUSIVE
0A32  72              :SAV1B   LD      (HL),D          ; STORE BLOCK COUNT
0A33  2A D5F1         :        LD      HL,(SPSAV)      ; H&L PT TO FILE LOC IN DIR
0A36  C3 DF0A         :        JP      SAVS            ; SAVE FILE
0A39  21 511C         :RNERR   LD      HL,MS51         ;'RANGE ERROR'
0A3C  C3 6603         :        JP      MESS
0A3F  2A D5F1         :SAVD    LD      HL,(SPSAV)      ; DELETE ENTRY IN DIR
0A42  0608            :        LD      B,8             ; DELETE NAME ONLY
0A44  3E20            :        LD      A,' '           ; <SP> FILL
0A46  77              :SAVD1   LD      (HL),A
0A47  23              :        INC     HL
0A48  10FC            :        DJNZ    SAVD1-$         ; NOW CREATE NEW DIR ENTRY
0A4A  21 00F3         :SAV2    LD      HL,DIRT         ; CREATE NEW DIR ENTRY
0A4D  0E10            :        LD      C,16            ; SKIP 16 BYTES EACH TIME
0A4F  7E              :SAV2L   LD      A,(HL)          ; GET CHAR
0A50  FE20            :        CP      ' '             ; <SP> MEANS FOUND NEW ENTRY LOC
0A52  2806            :        JR      Z,SAV2D-$
0A54  79              :        LD      A,C
0A55  CD 11D1         :        CALL    ADR             ; GET NEW ADR IN TABLE
0A58  18F5            :        JR      SAV2L-$
0A5A  E5              :SAV2D   PUSH    HL              ; H&L PT TO DIR ENTRY POINT
0A5B  21 00F3         :        LD      HL,DIRT         ; SCAN DIR FOR LARGEST DISK ADR
0A5E  3E40            :        LD      A,64            ; AT MOST 64 ENTRIES
0A60  32 15F0         :        LD      (LPCNT),A
0A63  01 0000         :        LD      BC,0            ; INITIAL VALUE OF 0
0A66  7E              :SAV21   LD      A,(HL)          ; GET CHAR
0A67  FE20            :        CP      ' '             ; SKIP IF NO ENTRY
0A69  2010            :        JR      NZ,SAV22-$
0A6B  3E10            :        LD      A,16            ; SKIP ENTRY
0A6D  CD 11D1         :SAV2A   CALL    ADR
0A70  3A 15F0         :        LD      A,(LPCNT)       ; DECR COUNT
0A73  3D              :        DEC     A
0A74  32 15F0         :        LD      (LPCNT),A
0A77  20ED            :        JR      NZ,SAV21-$
0A79  181F            :        JR      SAV23-$
0A7B  3E08            :SAV22   LD      A,LDSP
0A7D  CD 11D1         :        CALL    ADR             ; GET DISK ADR
0A80  5E              :        LD      E,(HL)          ; D&E=DISK ADR
0A81  23              :        INC     HL
0A82  56              :        LD      D,(HL)
0A83  78              :        LD      A,B             ; COMPARE AGAINST B&C
0A84  BA              :        CP      D
0A85  380A            :        JR      C,SVSL2-$
0A87  2004            :        JR      NZ,SVSL1-$
```

PROGRAM ERRORS -- 0

```
0A89  79              :        LD      A,C             ; D=B, SO COMPARE C AND E
0A8A  B8              :        CP      E
0A8B  3804            :        JR      C,SVSL2-$
0A8D  3E07            :SVSL1   LD      A,7             ; B&C LARGER, SO SKIP
0A8F  18DC            :        JR      SAV2A-$
0A91  4B              :SVSL2   LD      C,E             ; B&C=D&E
0A92  42              :        LD      B,D
0A93  2B              :        DEC     HL              ; SAVE NEW POINTER
0A94  22 E7F1         :        LD      (ADDS1),HL
0A97  23              :        INC     HL
0A98  18F3            :        JR      SVSL1-$         ; NOW SKIP
0A9A  2A E7F1         :SAV23   LD      HL,(ADDS1)      ; GET POINTER TO LARGEST ENTRY
0A9D  EB              :        EX      DE,HL           ; IN D&E
0A9E  C5              :        PUSH    BC              ; H&L=LARGEST DISK ADR
0A9F  E1              :        POP     HL
0AA0  EB              :        EX      DE,HL           ; D&E=LARGEST DISK ADR, H&L=POINTER
0AA1  23              :        INC     HL
0AA2  23              :        INC     HL
0AA3  7E              :        LD      A,(HL)          ; A=NO BLOCKS OF FILE AT LARGEST DISK ADR
0AA4  EB              :        EX      DE,HL           ; H&L=LARGEST DISK ADR, D&E=PTR
0AA5  CD 11D1         :        CALL    ADR             ; H&L=DISK ADR OF NEXT FREE BLOCK
0AA8  EB              :        EX      DE,HL           ; D&E=DISK ADR OF NEXT FREE BLOCK
0AA9  E1              :        POP     HL              ; H&L PT TO NEW DIR ENTRY
0AAA  22 E7F1         :        LD      (ADDS1),HL      ; ADDS1=FILE LOC IN DIR
0AAD  3E08            :        LD      A,LDSP          ; H&L NOW POINT TO DISK ADR ENTRY IN DIR ENTRY
0AAF  CD 11D1         :        CALL    ADR
0AB2  73              :        LD      (HL),E          ; STORE NEW DISK ADR
0AB3  23              :        INC     HL
0AB4  72              :        LD      (HL),D
0AB5  2A 10F0         :        LD      HL,(SRANG)      ; H=NO BLOCKS IN DISK FILE
0AB8  54              :        LD      D,H
0AB9  7D              :        LD      A,L             ; EXACT COUNT?
0ABA  B7              :        OR      A
0ABB  2801            :        JR      Z,SAV23A-$
0ABD  14              :        INC     D               ; D=NO BLOCKS IN DISK FILE, INCL
0ABE  2A E7F1         :SAV23A  LD      HL,(ADDS1)      ; H&L PT TO FILE ENTRY IN DIR
0AC1  3E0A            :        LD      A,LBDSP         ; STORE NO BLOCKS
0AC3  CD 11D1         :        CALL    ADR
0AC6  72              :        LD      (HL),D          ; STORE NO BLOCKS IN DISK FILE
0AC7  23              :        INC     HL
0AC8  3600            :        LD      (HL),0          ; 0 IS HIGH ORDER BLOCK COUNT
0ACA  23              :        INC     HL
0ACB  3600            :        LD      (HL),0          ; TYPE 0 FILE
0ACD  2A E7F1         :        LD      HL,(ADDS1)
0AD0  EB              :        EX      DE,HL           ; D&E PT TO FILE ENTRY IN DIR
0AD1  21 E9F1         :        LD      HL,FBUF         ; STORE FILE NAME
0AD4  0608            :        LD      B,NMLEN         ; NMLEN CHARS
0AD6  7E              :SAV24   LD      A,(HL)          ; GET CHAR
0AD7  12              :        LD      (DE),A          ; STORE CHAR
```

PROGRAM ERRORS -- 0

```
0AD8  23          :        INC     HL              ; INCR PTRS
0AD9  13          :        INC     DE
0ADA  10FA        :        DJNZ    SAV24-$
0ADC  2A E7F1     :        LD      HL,(ADDS1)      ; H&L PT TO FILE ENTRY IN DIR
0ADF  3E08        :SAVS    LD      A,LDSP          ; GET ADR OF FILE ON DISK
0AE1  CD 11D1     :        CALL    ADR             ; COMPUTE ADR
0AE4  5E          :        LD      E,(HL)          ; IN D&E
0AE5  23          :        INC     HL
0AE6  56          :        LD      D,(HL)
0AE7  23          :        INC     HL
0AE8  7E          :        LD      A,(HL)          ; A=NO BLOCKS
0AE9  2A 12F0     :        LD      HL,(SSTRT)      ;H&L=MEMORY START ADR
0AEC  0600        :        LD      B,0             ; WRITE
0AEE  CD DC08     :        CALL    DRIVE           ; GET DRIVE NUMBER
0AF1  EB          :        EX      DE,HL           ; D&E=MEMORY START ADR, H&L=DISK START ADR
0AF2  F5          :        PUSH    AF              ; SAVE ALL
0AF3  C5          :        PUSH    BC
0AF4  D5          :        PUSH    DE
0AF5  E5          :        PUSH    HL
0AF6  CD C302     :        CALL    DCOM
0AF9  E1          :        POP     HL              ; RESTORE ALL
0AFA  D1          :        POP     DE
0AFB  C1          :        POP     BC
0AFC  F1          :        POP     AF
0AFD  0602        :        LD      B,2             ; NOW VERIFY
0AFF  CD C302     :        CALL    DCOM
0B02  21 031C     :        LD      HL,MS33         ; PRINT MSG 1
0B05  CD 6C03     :        CALL    MESS1
0B08  21 0000     :RPDIR   LD      HL,0            ; SAVE DIR (START AT ADR 0 ON DISK)
0B0B  11 00F3     :        LD      DE,DIRT         ; FROM MEMORY
0B0E  3E04        :        LD      A,4             ; 4 BLOCKS
0B10  0600        :        LD      B,0             ; WRITE
0B12  CD DC08     :        CALL    DRIVE           ; GET DRIVE NUMBER
0B15  F5          :        PUSH    AF              ; SAVE REGS
0B16  C5          :        PUSH    BC
0B17  D5          :        PUSH    DE
0B18  E5          :        PUSH    HL
0B19  CD C302     :        CALL    DCOM
0B1C  E1          :        POP     HL
0B1D  D1          :        POP     DE
0B1E  C1          :        POP     BC
0B1F  F1          :        POP     AF
0B20  0602        :        LD      B,2             ; VERIFY
0B22  C3 C302     :        JP      DCOM
                  :*
                  :*  DSCAN SCANS THE CURRENT DISK DIRECTORY FOR THE FILE
                  :*    WHOSE NAME IS CONTAINED IN FBUF.  RETURNS W/ZERO SET
                  :*    AND HL PTS TO ENTRY IF FILE IS FOUND.
                  :*
```

PROGRAM ERRORS -- 0

```
0825  CD 560C          :DSCAN   CALL    DIR             ; GET DISK DIRECTORY
0828  21 00F3          :        LD      HL,DIRT
0828  0E40              :        LD      C,64            ; 64 POSSIBLE FILE ENTRIES
082D  11 E9F1          :DSCN1   LD      DE,FBUF
0830  0608              :        LD      B,NMLEN
0832  E5                :        PUSH    HL              ; BEGIN SCAN OF DIRT FOR FILE IN FBUF
0833  1A                :DSCNL   LD      A,(DE)          ; GET FILENAME IN FBUF
0834  BE                :        CP      (HL)            ; COMPARE AGAINST TABLE
0835  13                :        INC     DE
0836  23                :        INC     HL
0837  2009              :        JR      NZ,DSCNN-$      ; NO MATCH
0839  FE20              :        CP      ' '             ; <SP> MEANS FOUND ALSO
083B  2802              :        JR      Z,DSCNL1-$
083D  10F4              :        DJNZ    DSCNL-$         ; NMLEN CHARS?
083F  E1                :DSCNL1  POP     HL              ; FOUND; ... GET ADR
0840  AF                :        XOR     A               ; ZERO SET MEANS FOUND
0841  C9                :        RET
0842  E1                :DSCNN   POP     HL              ; NOT FOUND ... CORRECT ADR
0843  0D                :        DEC     C               ; DONE?
0844  2807              :        JR      Z,DSCNF-$
0846  3E10              :        LD      A,16            ; SKIP 16 BYTES IF NOT DONE
0848  CD 11D1           :        CALL    ADR
084B  18E0              :        JR      DSCN1-$         ; CONTINUE SCAN
084D  3E01              :DSCNF   LD      A,1             ; NOT ZERO IF NOT FOUND
084F  B7                :        OR      A
0850  C9                :        RET
```

PROGRAM ERRORS -- 0

```
                        :**** COMMAND -- ISRT
                        :*
                        :*  ISRT -- INSERT COMMAND.  THIS ALLOWS THE USER TO INSERT
                        :*       A GROUP OF LINES BEFORE ANY GIVEN LINE.  FORMAT
                        :*       ISRT LINENUMBER
                        :*
0851                    :INS     EQU     $
0851  CD A502             :      CALL    CKA1            ; CHECK FOR FIRST ARG
0854  CD D502             :      CALL    FIND            ; FIND THE LINE TO BE INSERTED BEFORE
0857  22 06F0           :INSE    LD      (INSNL),HL      ; SAVE NEXT LINE PTR
085A  EB                 :      EX      DE,HL           ; NOW IN D&E
085B  2A 9AF0            :      LD      HL,(EOFP)       ; GET PTR TO EOF
085E  22 08F0            :      LD      (INSLL),HL      ; SAVE POINTER TO END OF LAST LINE
0861  EB                 :      EX      DE,HL
0862  CD 9FD0            :      CALL    RANGE           ; COMPUTE RANGE
0865  ED43 10F0          :      LD      (SRANG),BC      ; SAVE RANGE
0869  CD 3FD0           :INSL    CALL    CRLF
086C  CD 5DD0            :      CALL    PCHAR           ; PRINT PROMPT
086F  3F                 :      DB      '?'
0870  CD 0101            :      CALL    READ            ; GET INPUT LINE
0873  21 01FE            :      LD      HL,IBUF         ; CHECK FOR CTRL-C AS FIRST
0876  7E                 :      LD      A,(HL)
0877  FE03               :      CP      3               ; ABORT IF FIRST CHAR
0879  2853               :      JR      Z,INSFX-$       ; FIX FILE
087B  0E00               :      LD      C,0             ; GET NO CHARS UP TO CTRL-C OR <CR>
087D  7E                 :INSL1   LD      A,(HL)
087E  FE03               :      CP      3               ; CTRL-C
0880  2808               :      JR      Z,INSL2-$
0882  FE0D               :      CP      0DH             ; <CR>
0884  2804               :      JR      Z,INSL2-$
0886  23                 :      INC     HL
0887  0C                 :      INC     C               ; INCR COUNT
0888  18F3               :      JR      INSL1-$
088A  3E07               :INSL2   LD      A,7             ; ADD 7 TO CHAR COUNT (1ST, LAST, + DIGITS
088C  81                 :      ADD     A,C
088D  32 00FE            :      LD      (IBUF-1),A      ; STORE NEW CHAR COUNT
0890  2A 08F0            :      LD      HL,(INSLL)      ; GET NEW END ADDR
0893  E5                 :      PUSH    HL
0894  CD 11D1            :      CALL    ADR             ; COMPUTE END ADDR
0897  22 08F0            :      LD      (INSLL),HL      ; NEW END
089A  D1                 :      POP     DE              ; D&E=OLD END
089B  ED4B 10F0          :      LD      BC,(SRANG)      ; B&C=NO BYTES TO MOVE
089F  CD 8DD0            :      CALL    RMOVL           ; MOVE TEXT BEYOND THE LINE TO INSERT
08A2  2A 06F0            :      LD      HL,(INSNL)      ; GET PTR TO OLD NEW LINE
08A5  11 00FE            :      LD      DE,IBUF-1       ; GET PTR TO NEW NEW LINE IN IBUF
08A8  1A                 :      LD      A,(DE)          ; GET CHAR COUNT
08A9  D606               :      SUB     6               ; ADJUST
08AB  4F                 :      LD      C,A             ; OMIT CHAR COUNT AND DIGITS NOW
08AC  1A                 :      LD      A,(DE)          ; GET CHAR COUNT

PROGRAM ERRORS --     0
```

```
0BAD  77              :        LD      (HL),A          ; STORE IT
0BAE  13              :        INC     DE
0BAF  23              :        INC     HL
0BB0  D5              :        PUSH    DE
0BB1  11 FDF1         :        LD      DE,ABUF         ; PT TO <LNUM>
0BB4  0604            :        LD      B,4             ; 4 DIGITS
0BB6  1A              :ISL3A   LD      A,(DE)          ; GET DIGIT CHARACTER
0BB7  77              :        LD      (HL),A          ; STORE DIGIT
0BB8  23              :        INC     HL
0BB9  13              :        INC     DE
0BBA  10FA            :        DJNZ    ISL3A-$
0BBC  D1              :        POP     DE
0BBD  3620            :        LD      (HL),' '        ; LAST SPACE
0BBF  23              :        INC     HL              ; PT TO DATA STORAGE
0BC0  CD 81D0         :        CALL    LMOVC           ; MOVE TEXT TO ITS NEW POSITION
0BC3  22 06F0         :        LD      (INSNL),HL      ; NEW LINE PTR
0BC6  2B              :        DEC     HL              ; CHECK FOR ENDING CTRL-C
0BC7  7E              :        LD      A,(HL)          ; CHECK LAST CHAR FOR CTRL-C
0BC8  FE03            :        CP      3
0BCA  209D            :        JR      NZ,INSL-$       ; CONTINUE IF NOT
0BCC  360D            :        LD      (HL),0DH        ; ENDING <CR> IF SO
0BCE  3A F9F1         :INSFX   LD      A,(SPCHAR)      ; CHECK FOR NO RENUMBER OPTION
0BD1  FE4E            :        CP      'N'
0BD3  C4 3811         :        CALL    NZ,RENME        ; RENUMBER IF NOT 'N'
                      :*
                      :*  FFIX -- FILE FIX ROUTINE.  OBTAIN THE EOFP AND MAXL
                      :*  OF THE PRIMARY FILE USING ONLY THE BOFP.
                      :*
0BD6                  :FFIX    EQU     $
0BD6  2A 98F0         :        LD      HL,(BOFP)       ; SCAN FILE FOR EOF
0BD9  CD 3F09         :        CALL    FCK             ; VERIFY FILE
0BDC  22 9AF0         :        LD      (EOFP),HL       ; SAVE EOFP
0BDF  2B              :FFIXN   DEC     HL              ; FIND MAXL
0BE0  15              :        DEC     D               ; DEC COUNT
0BE1  20FC            :        JR      NZ,FFIXN-$
0BE3  23              :        INC     HL
0BE4  11 9CF0         :        LD      DE,MAXL         ; SAVE MAXL
0BE7  0604            :        LD      B,4             ; 4 DIGITS
0BE9  7E              :FFIXN1  LD      A,(HL)          ; GET DIGIT
0BEA  12              :        LD      (DE),A          ; STORE DIGIT
0BEB  23              :        INC     HL
0BEC  13              :        INC     DE
0BED  10FA            :        DJNZ    FFIXN1-$
0BEF  C9              :        RET

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- APND
                        :*
                        :*  AUTO -- AUTOMATIC LINE NUMBERING COMMAND.
                        :*  THIS COMMAND ALLOWS THE USER TO ENTER STRINGS OF TEXT
                        :*  INTO A PRIMARY FILE WITHOUT TYPING LINE NUMBERS.  IT
                        :*  APPENDS THE INCOMING TEXT TO THE CURRENT PRIMARY FILE.
                        :*  WHEN DONE, THE ENTIRE FILE IS RENUMBERED (DEFAULT NUMBERING)
                        :*  AND PROCESSED
                        :*
0BF0                    :AUTO   EQU     $
0BF0  CD AF02           :       CALL    CKA11           ; CHECK FOR LINE NUMBER
0BF3  2812              :       JR      Z,AUTO2-$       ; APPEND TO END OF FILE IF NO LN
0BF5  CD D502           :       CALL    FIND            ; FIND SPECIFIED LINE; H&L PT TO CHAR COUNT
0BF8  7E                :       LD      A,(HL)          ; CHECK FOR EOF
0BF9  FE01              :       CP      1
0BFB  280A              :       JR      Z,AUTO2-$       ; DO APPEND TO FILE
0BFD  23                :       INC     HL              ; POINT TO FIRST VALID CHARACTER
0BFE  7E                :AUTO1  LD      A,(HL)          ; FIND <CR> FOR EOL
0BFF  23                :       INC     HL
0C00  FE0D              :       CP      0DH
0C02  20FA              :       JR      NZ,AUTO1-$
0C04  C3 570B           :       JP      INSE            ; DO INSERT AFTER THIS LINE
0C07  CD CD02           :AUTO2  CALL    FECHK           ; PRIMARY FILE MUST EXIST
0C0A  21 951C           :       LD      HL,LMAX         ; LAST LINE IN FILE
0C0D  11 FDF1           :       LD      DE,ABUF
0C10  01 0400           :       LD      BC,4
0C13  ED80              :       LDIR                    ; SET MAX LINE NUMBER
0C15  2A 9AF0           :       LD      HL,(EOFP)       ; POINT TO EOF
0C18  C3 570B           :       JP      INSE            ; DO INSERT AFTER EOF

PROGRAM ERRORS --    0
```

```
                        :**** COMMAND -- LNAME
                        :*
                        :*  RNME -- RENAME ANY FILE.  THE FORMAT OF THIS COMMAND
                        :*  IS   RNME FILE
                        :*    RNME RESPONDS WITH 'NEW FILE NAME?', TO WHICH THE
                        :*  USER TYPES A <CR> TO ABORT OR A STRING TO RENAME
                        :*    'LNAM' RENAMES THE LOCAL TEXT FILE; 'LNAMB' RENAMES
                        :*  THE LOCAL BINARY FILE
                        :*
0C1B                    :RNME    EQU     $
0C1B  F5                :        PUSH    AF              ; SAVE SPCHAR
0C1C  CD 9502           :        CALL    CKFN            ; CHECK FOR FILE NAME
0C1F  F1                :        POP     AF              ; GET SPCHAR
0C20  FE42              :        CP      'B'
0C22  CA FA12           :        JP      Z,BNAME
0C25  CD AF0E           :        CALL    FSEA            ; SCAN FOR FILE NAME
0C28  CA 6808           :        JP      Z,DRSDF         ; ERROR IF NOT FOUND
0C2B  E5                :RNMEE   PUSH    HL              ; RENAME ENTRY POINT
0C2C  0E08              :        LD      C,NMLEN         ; BLANK OUT 'NMLEN' CHARS
0C2E  CD DE12           :        CALL    CLERA           ; STORE BLANKS
0C31  E1                :        POP     HL
0C32  E5                :        PUSH    HL              ; SAVE ADR
0C33  CD 3A05           :        CALL    PNN             ; PRINT 'NEW NAME?' AND CHECK FOR <CR>
0C36  2002              :        JR      NZ,RNMES-$
0C38  E1                :        POP     HL              ; CLEAR STACK
0C39  C9                :        RET
0C3A  D1                :RNMES   POP     DE              ; D&E=ADR IN TABLE
0C3B  7E                :RNMS1   LD      A,(HL)          ; GET CHAR
0C3C  FE20              :        CP      ' '             ; DONE IF SPACE
0C3E  C8                :        RET     Z
0C3F  FE0D              :        CP      0DH             ; DONE IF <CR>
0C41  C8                :        RET     Z
0C42  CD 05D1           :        CALL    CAPS            ; CONVERT LOWER CASE TO UPPER
0C45  12                :        LD      (DE),A          ; SAVE CHAR
0C46  23                :        INC     HL
0C47  13                :        INC     DE
0C48  0D                :        DEC     C
0C49  20F0              :        JR      NZ,RNMS1-$
0C4B  C9                :        RET

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- LDIR
                        :*
                        :*  LDIR -- LOCAL FILE DIRECTORY
                        :*
 0C4C                   :LDIR    EQU     $
 0C4C  FE42             :        CP      'B'             ; BINARY FILE DIRECTORY
 0C4E  CA 0913          :        JP      Z,BDIR
 0C51  0E0A             :        LD      C,MAXFIL
 0C53  C3 1A0E          :        JP      FOUL

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- DDIR
                        :*
                        :*  DIRD -- DISK DIRECTORY COMMAND
                        :*
0C56                    :DIR    EQU     $
0C56  CD CD08           :       CALL    DRIVEC          ; CHECK FOR DRIVE NUMBER
0C59  3E04              :       LD      A,4             ; 4 BLOCKS OF DIRECTORY TO LOAD
0C5B  0601              :       LD      B,1             ; READ
0C5D  CD DC08           :       CALL    DRIVE           ; GET DRIVE NUMBER
0C60  11 00F3           :       LD      DE,DIRT         ; INTO TABLE (1K)
0C63  21 0000           :       LD      HL,0            ; START AT BLOCK ZERO
0C66  C3 C302           :       JP      DCOM            ; FDOS DISK UTILITY
0C69  CD 560C           :DIRD   CALL    DIR
0C6C  21 00F3           :       LD      HL,DIRT
0C6F  3E40              :       LD      A,64            ; LOOK AT 64 ENTRIES
0C71  0EFF              :       LD      C,0FFH          ; SET PAGING CNT
0C73  32 14F0           :       LD      (LCTRL),A       ; SAVE
0C76  CD A903           :       CALL    LINIT           ; SET LINE COUNT
0C79  7E                :DIRPL  LD      A,(HL)          ; GET 1ST CHAR
0C7A  FE20              :       CP      ' '             ; <SP>?
0C7C  200F              :       JR      NZ,DIRP2-$
0C7E  3E10              :       LD      A,16            ; SKIP 16 BYTES
0C80  CD 11D1           :       CALL    ADR
0C83  3A 14F0           :       LD      A,(LCTRL)       ; DONE?
0C86  3D                :       DEC     A
0C87  32 14F0           :       LD      (LCTRL),A
0C8A  C8                :       RET     Z               ; DONE!
0C8B  18EC              :       JR      DIRPL-$
0C8D  1608              :DIRP2  LD      D,8             ; PRINT 8 CHARS
0C8F  0606              :       LD      B,6             ; 6 BLANKS BETWEEN ENTRIES
0C91  CD 42D0           :DIRP2A CALL    BLK1
0C94  10FB              :       DJNZ    DIRP2A-$
0C96  0C                :       INC     C               ; INCR CNT
0C97  79                :       LD      A,C
0C98  E601              :       AND     1               ; <CR> ON EVERY OTHER
0C9A  2009              :       JR      NZ,DIRP3-$
0C9C  CD 9603           :       CALL    LINECNT         ; PAGE
0C9F  CD 3FD0           :       CALL    CRLF
0CA2  CD F400           :       CALL    INK             ; CHECK FOR INTERRUPT
0CA5  46                :DIRP3  LD      B,(HL)          ; GET CHAR
0CA6  CD 24D0           :       CALL    OUT8
0CA9  23                :       INC     HL
0CAA  15                :       DEC     D
0CAB  20F8              :       JR      NZ,DIRP3-$
0CAD  23                :       INC     HL
0CAE  23                :       INC     HL
0CAF  CD 42D0           :       CALL    BLK1            ; PRINT <SP>
0CB2  7E                :       LD      A,(HL)          ; GET NUMBER OF BLOCKS
0CB3  CD 69D0           :       CALL    DOUT            ; PRINT IN DECIMAL
```

PROGRAM ERRORS -- 0

```
0CB6  23              :        INC     HL
0CB7  23              :        INC     HL
0CB8  CD 42D0         :        CALL    BLK1            ; PRINT <SP>
0CBB  7E              :        LD      A,(HL)          ; GET FILE TYPE
0CBC  C630            :        ADD     A,'0'           ; CONVERT TO ASCII
0CBE  CD 1ED0         :        CALL    OUTPUT
0CC1  FE31            :        CP      '1'             ; TYPE 1 FILE?
0CC3  2018            :        JR      NZ,DIRP5-$
0CC5  23              :        INC     HL
0CC6  7E              :        LD      A,(HL)          ; GET EXEC ADR
0CC7  23              :        INC     HL
0CC8  FE20            :        CP      ' '
0CCA  2805            :        JR      Z,DIRP4-$       ; SKIP IF <SP>
0CCC  5F              :        LD      E,A             ; GET ADR
0CCD  56              :        LD      D,(HL)
0CCE  CD E80C         :        CALL    PDEP
0CD1  23              :DIRP4   INC     HL
0CD2  23              :        INC     HL
0CD3  3A 14F0         :        LD      A,(LCTRL)       ; DONE?
0CD6  3D              :        DEC     A
0CD7  32 14F0         :        LD      (LCTRL),A
0CDA  C8              :        RET     Z
0CDB  189C            :        JR      DIRPL-$
0CDD  23              :DIRP5   INC     HL              ; SKIP ADDRESS
0CDE  23              :        INC     HL
0CDF  0605            :        LD      B,5             ; 5 <SP>
0CE1  CD 42D0         :DIRP6   CALL    BLK1
0CE4  10FB            :        DJNZ    DIRP6-$
0CE6  18E9            :        JR      DIRP4-$
0CE8  CD 42D0         :PDEP    CALL    BLK1            ; PRINT LEADING <SP>
0CEB  C3 75D0         :        JP      PDE             ; PRINT DE AS HEX

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- LDEL
                        :*
                        :*  FDEL -- DELETE ENTRY IN LOCAL FILE:
                        :*    THE BINARY FILE DIRECTORY ENTRY IS DELETED IF THE
                        :*    DELETE COMMAND IS OF THE FORM 'LDELB', WHERE 'LDEL'
                        :*    DELETES JUST THE TEXT FILE
                        :*
0CEE                    :FDEL   EQU     $
0CEE  F5                :       PUSH    AF              ; SAVE SPCHAR
0CEF  CD 9502           :       CALL    CKFN            ; CHECK FOR FILE NAME
0CF2  F1                :       POP     AF              ; GET SPCHAR & CHECK FOR BINARY
0CF3  FE42              :       CP      'B'
0CF5  CA EA12           :       JP      Z,BDEL
0CF8  CD AF0E           :       CALL    FSEA            ; SCAN DIRECTORY
0CFB  CA 6808           :       JP      Z,DRSDF
0CFE  11 90F0           :       LD      DE,FILE0        ; H&L PT TO FILE, D&E PT TO PRIMARY
0D01  7B                :       LD      A,E
0D02  BD                :       CP      L
0D03  2820              :       JR      Z,FDELP-$       ; IF SAME, DELETE PRIMARY
0D05  E5                :       PUSH    HL
0D06  11 A0F0           :       LD      DE,FILE0+FELEN
0D09  7D                :       LD      A,L
0D0A  93                :       SUB     E
0D0B  47                :       LD      B,A             ; B=NO OF BYTES TO MOVE
0D0C  E1                :       POP     HL              ; RESTORE PTR
0D0D  E5                :       PUSH    HL
0D0E  11 0F00           :       LD      DE,FELEN-1      ; PT TO LAST BYTE
0D11  19                :       ADD     HL,DE           ; H&L PT TO BYTE
0D12  D1                :       POP     DE              ; D&E PT TO 1ST BYTE TO MOVE
0D13  1B                :       DEC     DE
0D14  1A                :FDELL  LD      A,(DE)          ; DO MOVE
0D15  77                :       LD      (HL),A
0D16  2B                :       DEC     HL
0D17  1B                :       DEC     DE
0D18  10FA              :       DJNZ    FDELL-$
0D1A  21 A0F0           :       LD      HL,FILE0+FELEN  ; ZERO 1ST FILE
0D1D  0610              :       LD      B,FELEN
0D1F  AF                :       XOR     A               ; A=0
0D20  77                :FDL1   LD      (HL),A          ; DO DELETE
0D21  23                :       INC     HL
0D22  10FC              :       DJNZ    FDL1-$
0D24  C9                :       RET
0D25  11 1000           :FDELP  LD      DE,FELEN        ; DELETE PRIMARY
0D28  21 A0F0           :       LD      HL,FILE0+FELEN  ; SEARCH FOR NEW PRIMARY
0D2B  0609              :       LD      B,MAXFIL-1
0D2D  7E                :FDLP1  LD      A,(HL)          ; FIND NON-ZERO
0D2E  B7                :       OR      A
0D2F  200D              :       JR      NZ,FDLPF-$
0D31  19                :       ADD     HL,DE           ; PT TO NEXT ENTRY
```

PROGRAM ERRORS -- 0

```
0D32  10F9          :        DJNZ    FDLP1-$
0D34  21 90F0       :        LD      HL,FILE0        ; NONE FOUND
0D37  AF            :        XOR     A               ; ZERO PRIMARY
0D38  77            :FDLP2   LD      (HL),A
0D39  23            :        INC     HL
0D3A  1D            :        DEC     E
0D3B  20FB          :        JR      NZ,FDLP2-$
0D3D  C9            :        RET
0D3E  E5            :FDLPF   PUSH    HL              ; CREATE NEW PRIMARY
0D3F  43            :        LD      B,E             ; C=B=NO BYTES IN ENTRY
0D40  4B            :        LD      C,E
0D41  11 90F0       :        LD      DE,FILE0
0D44  7E            :FDLF1   LD      A,(HL)          ; MOVE TO PRIMARY
0D45  12            :        LD      (DE),A
0D46  13            :        INC     DE
0D47  23            :        INC     HL
0D48  10FA          :        DJNZ    FDLF1-$
0D4A  E1            :        POP     HL              ; DELETE OMOVENTRY
0D4B  AF            :        XOR     A               ; A=0
0D4C  59            :        LD      E,C
0D4D  18E9          :        JR      FDLP2-$         ; DO DELETION

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- TABS
                        :*
                        :*  TABST -- CONTROL POINT FOR TAB SET, EXAMINE, AND RESET
                        :*
 0D4F                   :TABST   EQU     $
 0D4F  FE4C             :        CP      'L'             ; LIST?
 0D51  CA A5D0          :        JP      Z,TABE          ; DO TAB EXAMINE
 0D54  FE52             :        CP      'R'             ; RESET?
 0D56  CA A8D0          :        JP      Z,TABR          ; DO TAB RESET
 0D59  C3 A2D0          :        JP      TABS            ; DO TAB SET OTHERWISE

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- FILE
                        :*
                        :* THIS ROUTINE INITIALIZES THE BEGINNING OF FILE ADDRESS
                        :*  AND END OF FILE ADDRESS AS WELL AS THE FILE AREA
                        :*  WHEN THE FILE COMMAND IS USED.
                        :*
0D5C                    :FILE   EQU     $
0D5C  FE42              :       CP      'B'
0D5E  CA A012           :       JP      Z,FILEB
0D61  3A E9F1           :       LD      A,(FBUF)
0D64  B7                :       OR      A
0D65  CA 180E           :       JP      Z,FPRMP         ; PRINT PRIMARY FILE INFO
0D68  CD AF0E           :       CALL    FSEA            ; LOOK UP FILE
0D6B  EB                :       EX      DE,HL           ; PNTR IN DE
0D6C  2019              :       JR      NZ,FENTF-$
                        :*  NO ENTRY IN LOCAL FILE DIRECTORY
0D6E  3A FDF1           :       LD      A,(ABUF)        ; CHECK FOR PARAM
0D71  B7                :       OR      A
0D72  200A              :       JR      NZ,FILE1-$      ; NO?? -- LOCATE IT
0D74  CD 680E           :       CALL    WNEXT           ; H&L=ADR OF NEXT AVAILABLE BYTE OF WS
0D77  22 0DF2           :       LD      (BBUF),HL       ; SET UP ADR
0D7A  7C                :       LD      A,H
0D7B  32 FDF1           :       LD      (ABUF),A        ; NON-ZERO FOR ASCII OF ADR
                        :*  CHECK FOR ROOM IN DIRECTORY
0D7E  3A F4F1           :FILE1  LD      A,(FEF)
0D81  B7                :       OR      A
0D82  2016              :       JR      NZ,FSP-$
0D84  C3 2605           :       JP      CUSE1           ; DIR FULL ERROR MSG
                        :*  ENTRY FOUND ARE THESE PARAMETERS
0D87  3A FDF1           :FENTF  LD      A,(ABUF)
0D8A  B7                :       OR      A
0D8B  281C              :       JR      Z,FPRM-$
0D8D  2A 0DF2           :       LD      HL,(BBUF)
0D90  7C                :       LD      A,H
0D91  B5                :       OR      L
0D92  2815              :       JR      Z,FPRM-$
0D94  21 371C           :       LD      HL,MS39         ; NO-NO CAN'T DO
0D97  C3 6603           :       JP      MESS            ;  IT -- DELETE FIRST
                        :*  MOVE FILE NAME TO BLOCK POINTED TO BY FREAD
0D9A  2A F2F1           :FSP    LD      HL,(FREAD)
0D9D  E5                :       PUSH    HL
0D9E  11 E9F1           :       LD      DE,FBUF         ; FILE NAME POINTER IN H,L
0DA1  0E08              :       LD      C,NMLEN         ; NAME LENGTH COUNT
0DA3  0600              :       LD      B,0
0DA5  CD 8400           :       CALL    LMOVL           ; MOVE IT
0DA8  D1                :       POP     DE              ; RESTORE ENTRY POINTER
                        :*  MAKE FILE POINTED TO BY D,E PRIMARY
0DA9  21 90F0           :FPRM   LD      HL,FILE0
0DAC  0E10              :       LD      C,FELEN         ; ENTRY LENGTH

PROGRAM ERRORS --    0
```

```
0DAE  1A          :FPRM1   LD      A,(DE)
0DAF  46          :        LD      B,(HL)
0DB0  77          :        LD      (HL),A          ; EXCHANGE
0DB1  78          :        LD      A,B
0DB2  12          :        LD      (DE),A
0DB3  13          :        INC     DE
0DB4  23          :        INC     HL              ; BUMP POINTERS
0DB5  0D          :        DEC     C               ; TEST COUNT
0DB6  20F6        :        JR      NZ,FPRM1-$
                  :*
                  :*  PUT PRIMARY FILE AT END OF WORKSPACE
                  :*
0DB8  CD 6B0E     :        CALL    WNEXT           ; GET NEXT AVAILABLE ADDRESS IN WS
0DBB  22 04F0     :        LD      (WSBF),HL       ; STORE IN WS BUFFER
0DBE  2A 9AF0     :        LD      HL,(EOFP)       ; COMPUTE NUMBER OF BYTES TO MOVE
0DC1  EB          :        EX      DE,HL
0DC2  2A 98F0     :        LD      HL,(BOFP)       ; H&L=BOFP, D&E=EOFP
0DC5  CD 9FD0     :        CALL    RANGE           ; B&C=NO BYTES
0DC8  2A 04F0     :        LD      HL,(WSBF)       ; COMPUTE NEW END ADDRESS
0DCB  09          :        ADD     HL,BC           ; H&L=NEW NEXT AVAILABLE BYTE
0DCC  EB          :        EX      DE,HL
0DCD  2A 02F0     :        LD      HL,(WSPE)       ; CHECK AGAINST END OF WS
0DD0  CD 5F03     :        CALL    CHLDE           ; COMPARE H&L AGAINST D&E
0DD3  300E        :        JR      NC,WSOK-$       ; D&E<H&L
0DD5  21 8A1C     :        LD      HL,MS93         ; 'WS BOUNDARY EXCEEDED'
0DD8  C5          :        PUSH    BC              ; SAVE RANGE
0DD9  CD 8A03     :        CALL    SCRN
0DDC  CD 78D0     :        CALL    PQ1             ; GET RESPONSE
0DDF  C1          :        POP     BC
0DE0  CA 8A00     :        JP      Z,EOR           ; ABORT
0DE3  2A 98F0     :WSOK    LD      HL,(BOFP)       ; MOVE FROM CURRENT BOF
0DE6  EB          :        EX      DE,HL
0DE7  2A 04F0     :        LD      HL,(WSBF)       ; TO NEXT AVAILABLE
0DEA  C5          :        PUSH    BC
0DEB  CD 84D0     :        CALL    LMOVL           ; MOVE B&C BYTES
0DEE  C1          :        POP     BC
0DEF  2A 04F0     :        LD      HL,(WSBF)       ; SET NEW BOF
0DF2  22 98F0     :        LD      (BOFP),HL
0DF5  09          :        ADD     HL,BC           ; SET NEW EOF
0DF6  2B          :        DEC     HL              ; ONE LESS THAN RANGE
0DF7  22 9AF0     :        LD      (EOFP),HL
                  :*   COLLECT GARBAGE (PACK FILES TOGETHER)
0DFA  2A 00F0     :        LD      HL,(WSPS)       ; GET START OF WS
0DFD  22 04F0     :        LD      (WSBF),HL       ; INITIALIZE NEXT AVAILABLE BYTE
0E00  CD ED0E     :        CALL    PACK1           ; PACK IT
                  :*  CHECK FOR ADDRESS PARAMETER
0E03  3A FDF1     :        LD      A,(ABUF)
0E06  B7          :        OR      A
0E07  280F        :        JR      Z,FPRMP-$       ; NO ADDRESS PARAMETER
```

PROGRAM ERRORS -- 0

```
                        :* PROCESS ADDRESS PARAMETER
0E09  2A 0DF2           :       LD      HL,(BBUF)       ; GET ADDRESS
0E0C  22 98F0           :       LD      (BOFP),HL       ; SET BEGIN
0E0F  22 9AF0           :       LD      (EOFP),HL       ; SET END
0E12  3601              :       LD      (HL),1          ; NON-ZERO -- SET EOF
0E14  AF                :       XOR     A               ; AND MAX LINE NO
0E15  32 9CF0           :       LD      (MAXL),A
0E18  0E01              :FPRMP  LD      C,1
                        :*  OUTPUT THE NO OF ENTRIES IN C
0E1A  21 90F0           :FOUL   LD      HL,FILE0
0E1D  79                :       LD      A,C
0E1E  32 F4F1           :FOUTL  LD      (FOCNT),A       ; SAVE COUNT
0E21  E5                :       PUSH    HL
0E22  11 0800           :       LD      DE,NMLEN
0E25  19                :       ADD     HL,DE
0E26  7E                :       LD      A,(HL)
0E27  B7                :       OR      A
0E28  2008              :       JR      NZ,FPRI-$       ; NON ZERO, OK TO OUTPUT
0E2A  23                :       INC     HL
0E2B  86                :       ADD     A,(HL)
0E2C  23                :       INC     HL
0E2D  2006              :       JR      NZ,FPRI-$
0E2F  33                :       INC     SP
0E30  33                :       INC     SP
0E31  23                :       INC     HL
0E32  23                :       INC     HL
0E33  1822              :       JR      FEET-$
                        :*  HAVE AN ENTRY TO OUTPUT
0E35  E1                :FPRI   POP     HL              ; PTR
0E36  0E08              :       LD      C,NMLEN
0E38  CD 3FD0           :       CALL    CRLF
0E3B  46                :FPRI1  LD      B,(HL)          ; LOAD CHARACTER TO B
0E3C  CD 24D0           :       CALL    OUT8
0E3F  0D                :       DEC     C
0E40  23                :       INC     HL
0E41  20F8              :       JR      NZ,FPRI1-$      ; DO THE REST
                        :*  NOW OUTPUT BEGIN-END PTRS
0E43  CD 620E           :       CALL    FPTRP           ; OUTPUT BEGIN
0E46  CD 620E           :       CALL    FPTRP           ; OUTPUT END
0E49  E5                :       PUSH    HL              ; SAVE PTR TO LMAX
                        :*  OUTPUT MAXL VALUE
0E4A  0604              :       LD      B,4             ; 4 CHARS
0E4C  CD 42D0           :       CALL    BLK1            ; PRINT <SP>
0E4F  7E                :FPRI2  LD      A,(HL)          ; GET CHAR
0E50  CD 1ED0           :       CALL    OUTPUT          ; PRINT
0E53  23                :       INC     HL              ; PT TO NEXT
0E54  10F9              :       DJNZ    FPRI2-$
0E56  E1                :       POP     HL              ; GET HL PTR
                        :*  TEST COUNT, H,L POINTS PAST EOFP

PROGRAM ERRORS --     0
```

```
0E57  11 0400          :FEET   LD      DE,FELEN-NMLEN-4
0E5A  19               :       ADD     HL,DE           ; MOVE TO NEXT ENTRY
0E5B  3A F4F1          :       LD      A,(FOCNT)
0E5E  3D               :       DEC     A               ; TEST COUNT
0E5F  208D             :       JR      NZ,FOUTL-$      ; MORE TO DO
0E61  C9               :       RET                     ; DONE !
                       :*  OUTPUT NUMBER POINTED TO BY H,L; ON RET, H,L POINT 2 WORDS LATER
0E62  CD 42D0          :FPTRP  CALL    BLK1            ; SPACE
0E65  D5               :       PUSH    DE              ; SAVE DE
0E66  CD 5903          :       CALL    PEDADR          ; PRINT MEMORY PTED TO BY HL AS HEX; STORED LOW,
0E69  D1               :       POP     DE              ; RESTORE DE
0E6A  C9               :       RET
                       :*
                       :*  DETERMINE NEXT AVAILABLE FILE ADR IN WORKSPACE
                       :*
0E6B  2A 00F0          :WNEXT  LD      HL,(WSPS)       ; GET PTR TO START OF WS
0E6E  2B               :       DEC     HL
0E6F  22 04F0          :       LD      (WSBF),HL       ; STORE PTR
0E72  21 90F0          :       LD      HL,FILE0        ; PT TO FILE DIR
0E75  3E0A             :       LD      A,MAXFIL        ; CHECK ENTRIES
0E77  32 15F0          :       LD      (LPCNT),A
0E7A  E5               :WNXT   PUSH    HL              ; SAVE ENTRY PTR
0E7B  7E               :       LD      A,(HL)          ; CHECK FOR FN
0E7C  B7               :       OR      A
0E7D  2815             :       JR      Z,WNXE-$
0E7F  3E0A             :       LD      A,NMLEN+2       ; GET EOFP
0E81  CD 11D1          :       CALL    ADR
0E84  5E               :       LD      E,(HL)          ; IN D&E
0E85  23               :       INC     HL
0E86  56               :       LD      D,(HL)
0E87  2A 04F0          :       LD      HL,(WSBF)       ; COMPARE
0E8A  7C               :       LD      A,H
0E8B  BA               :       CP      D
0E8C  381A             :       JR      C,WEX-$
0E8E  2004             :       JR      NZ,WNXE-$
0E90  7D               :       LD      A,L
0E91  BB               :       CP      E
0E92  3814             :       JR      C,WEX-$
0E94  E1               :WNXE   POP     HL              ; GET PTR
0E95  3E10             :       LD      A,NMLEN+8       ; PT TO NEXT ENTRY
0E97  CD 11D1          :       CALL    ADR
0E9A  3A 15F0          :       LD      A,(LPCNT)       ; ALL ENTRIES CHECKED?
0E9D  3D               :       DEC     A
0E9E  32 15F0          :       LD      (LPCNT),A
0EA1  20D7             :       JR      NZ,WNXT-$
0EA3  2A 04F0          :       LD      HL,(WSBF)       ; PT TO NEXT AVAILABLE SPACE
0EA6  23               :       INC     HL
0EA7  C9               :       RET
0EA8  6B               :WEX    LD      L,E             ; EXCHANGE FOR NEXT HIGHER PTR
```

PROGRAM ERRORS -- 0

```
0EA9  62              :        LD      H,D
0EAA  22 04F0         :        LD      (WSBF),HL
0EAD  18E5            :        JR      WNXE-$
                      :*
                      :* SEARCH THE FILE DIRECTORY FOR THE FILE
                      :*  WHOSE NAME IN IN FBUF.
                      :*
                      :* UPON EXIT, IF THE FILE WAS FOUND, THE NZ FLAG IS ON
                      :*  AND H&L POINT TO THE DIRECTORY ENTRY FOR THE SPECIFIED
                      :*  FILE; IF FILE WAS NOT FOUND, THE Z FLAG IS ON AND THE
                      :*  FREE ENTRY FLAG (FEF) IS NON-ZERO (THERE IS A FREE
                      :*  ENTRY) OR ZERO (NO FREE ENTRY) INDICATING IF THERE IS ROOM IN THE
                      :*  LOCAL FILE DIRECTORY FOR ANOTHER FILE ENTRY; ALSO,
                      :*  IF THERE IS ROOM FOR ANOTHER FILE ENTRY (FEF IS NON-ZERO),
                      :*  FREAD POINTS TO THE AVAILABLE DIRECTORY ENTRY LOCATION
                      :*  IN WHICH A NEW FILE MAY BE PLACED
                      :*
0EAF  AF              :FSEA    XOR     A
0EB0  32 F4F1         :        LD      (FEF),A         ; CLAIM NO FREE ENTRIES
0EB3  060A            :        LD      B,MAXFIL        ; COUNT OF ENTRIES
0EB5  11 90F0         :        LD      DE,FILE0        ; TABLE ADDRESS
0EB8  21 E9F1         :FSEA1   LD      HL,FBUF
0EBB  0E08            :        LD      C,NMLEN
0EBD  CD 4201         :        CALL    SEAR            ; TEST STRINGS
0EC0  F5              :        PUSH    AF              ; SAVE FLAG
0EC1  D5              :        PUSH    DE
0EC2  1A              :        LD      A,(DE)          ; GET BOFP
0EC3  B7              :        OR      A               ; EMPTY ENTRY?
0EC4  201C            :        JR      NZ,FSEA3-$
0EC6  13              :        INC     DE              ; STORE OTHER WORD
0EC7  1A              :        LD      A,(DE)
0EC8  B7              :        OR      A
0EC9  2017            :        JR      NZ,FSEA3-$      ; NOPE -- GO TEST FOR MATCH
0ECB  EB              :        EX      DE,HL
0ECC  11 F7FF         :        LD      DE,0-NMLEN-1
0ECF  19              :        ADD     HL,DE           ; MOVE TO BEGINNING
0ED0  22 F2F1         :        LD      (FREAD),HL      ; SAVE ADDRESS
0ED3  7A              :        LD      A,D
0ED4  32 F4F1         :        LD      (FEF),A         ; SET FREE ENTRY FOUND
0ED7  E1              :        POP     HL              ; RESTORE INTERIM PTR
0ED8  F1              :        POP     AF              ; UNJUNK STACK
                      :*  MOVE TO NEXT ENTRY
0ED9  11 0800         :FSEA2   LD      DE,FELEN-NMLEN
0EDC  19              :        ADD     HL,DE
0EDD  EB              :        EX      DE,HL           ; NEXT ENTRY ADDRESS IN DE
0EDE  05              :        DEC     B               ; TEST COUNT
0EDF  C8              :        RET     Z               ; DONE -- NOPE
0EE0  18D6            :        JR      FSEA1-$         ; TRY NEXT
                      :*  ENTRY WASN'T FREE; TEST FOR MATCH

PROGRAM ERRORS --     0
```

```
0EE2  E1              :FSEA3   POP     HL
0EE3  F1              :        POP     AF
0EE4  20F3            :        JR      NZ,FSEA2-$      ; IF ZERO CLEAR, NO MATCH
                      :*  ENTRY FOUND
0EE6  11 F8FF         :        LD      DE,0-NMLEN      ; BACKUP
0EE9  19              :        ADD     HL,DE           ; H,L POINTS TO ENTRY
0EEA  7A              :        LD      A,D
0EEB  87              :        OR      A               ; CLEAR ZERO
0EEC  C9              :        RET                     ; THAT'S ALL
                      :
                      :
                      :
                      :*
                      :*  PACK -- GARBAGE COLLECTION (PACK LOCAL FILES TOGETHER)
                      :*
0EED  21 FFFF         :PACK1   LD      HL,0FFFFH       ; SET CURRENT ADR
0EF0  22 06F0         :        LD      (FCUR),HL
0EF3  21 90F0         :        LD      HL,FILE0        ; PT TO LOCAL FILE DIR
0EF6  060A            :        LD      B,MAXFIL        ; NO FILES
0EF8  AF              :        XOR     A               ; SET PACK KEY
0EF9  32 71F2         :        LD      (PKEY),A
0EFC  7E              :FCPI    LD      A,(HL)          ; GET 1ST CHAR OF FN
0EFD  E5              :        PUSH    HL              ; SAVE PTR
0EFE  B7              :        OR      A
0EFF  204B            :        JR      NZ,FCPIL-$      ; NAME PRESENT -- COMPRESS IF NECESSARY
0F01  E1              :FCPI1   POP     HL              ; RESTORE PTR
0F02  3E10            :FCPI2   LD      A,NMLEN+8       ; PT TO NEXT ENTRY IN DIR
0F04  CD 11D1         :        CALL    ADR
0F07  10F3            :        DJNZ    FCPI-$          ; DECR COUNT OF FILE CHECKED
0F09  3A 71F2         :        LD      A,(PKEY)        ; FILE FOUND?
0F0C  B7              :        OR      A
0F0D  C8              :        RET     Z               ; DONE IF NOT
0F0E  2A 08F0         :        LD      HL,(FCURE)      ; PT TO ENTRY OF FILE TO MOVE
0F11  3E08            :        LD      A,NMLEN         ; GET ITS BOFP
0F13  CD 11D1         :        CALL    ADR
0F16  5E              :        LD      E,(HL)
0F17  23              :        INC     HL
0F18  56              :        LD      D,(HL)
0F19  23              :        INC     HL
0F1A  EB              :        EX      DE,HL
0F1B  22 E7F1         :        LD      (ADDS1),HL      ; SAVE TEMPORRAILY
0F1E  EB              :        EX      DE,HL
0F1F  5E              :        LD      E,(HL)          ; GET ITS EOFP
0F20  23              :        INC     HL
0F21  56              :        LD      D,(HL)
0F22  2A E7F1         :        LD      HL,(ADDS1)      ; H&L=BOFP, D&E=EOFP
0F25  CD 9FD0         :        CALL    RANGE           ; B&C=NO BYTES
0F28  EB              :        EX      DE,HL
0F29  2A 04F0         :        LD      HL,(WSBF)       ; H&L=LOC TO MOVE TO, D&E=LOC TO MOVE FROM

PROGRAM ERRORS --     0
```

```
0F2C  CD 84D0          :        CALL    LMOVL           ; MOVE FILE
0F2F  E5               :        PUSH    HL              ; SAVE PTR TO NEXT AVAILABLE ADR
0F30  2A 08F0          :        LD      HL,(FCURE)      ; RESET BOFP AND EOFP OF MOVED FILE
0F33  3E08             :        LD      A,NMLEN
0F35  CD 11D1          :        CALL    ADR
0F38  ED5B 04F0        :        LD      DE,(WSBF)       ; NEW BOFP
0F3C  73               :        LD      (HL),E          ; STORE BOFP
0F3D  23               :        INC     HL
0F3E  72               :        LD      (HL),D
0F3F  23               :        INC     HL
0F40  D1               :        POP     DE              ; GET NEXT AVAILABLE ADR
0F41  1B               :        DEC     DE              ; NEW EOFP
0F42  73               :        LD      (HL),E          ; STORE EOFP
0F43  23               :        INC     HL
0F44  72               :        LD      (HL),D
0F45  13               :        INC     DE
0F46  EB               :        EX      DE,HL
0F47  22 04F0          :        LD      (WSBF),HL       ; NEW NEXT AVAILABLE ADR
0F4A  18A1             :        JR      PACK1-$
0F4C  3E08             :FCPIL   LD      A,NMLEN         ; GET BOFP
0F4E  CD 11D1          :        CALL    ADR
0F51  5E               :        LD      E,(HL)
0F52  23               :        INC     HL
0F53  56               :        LD      D,(HL)          ; D&E=BOFP
0F54  2A 04F0          :        LD      HL,(WSBF)       ; CHECK AGAINST NEXT AVAILABLE
0F57  CD 5F03          :        CALL    CHLDE
0F5A  280A             :        JR      Z,FCML1-$       ; MOVE IT IF ZERO
0F5C  30A3             :        JR      NC,FCPI1-$      ; ABORT IF H&L>D&E
0F5E  2A 06F0          :        LD      HL,(FCUR)       ;CHECK AGAINST CURRENT FILE TO MOVE
0F61  CD 5F03          :        CALL    CHLDE
0F64  389B             :        JR      C,FCPI1-$       ; ABORT IF CURRENT<CONSIDERED FILE
0F66  E1               :FCML1   POP     HL              ; PT TO ENTRY
0F67  ED53 06F0        :        LD      (FCUR),DE       ; NEW CURRENT FILE
0F6B  22 08F0          :        LD      (FCURE),HL      ; NEW PTR
0F6E  3E01             :        LD      A,1             ; SET PROCESS KEY
0F70  32 71F2          :        LD      (PKEY),A
0F73  C3 020F          :        JP      FCPI2

PROGRAM ERRORS --      0
```

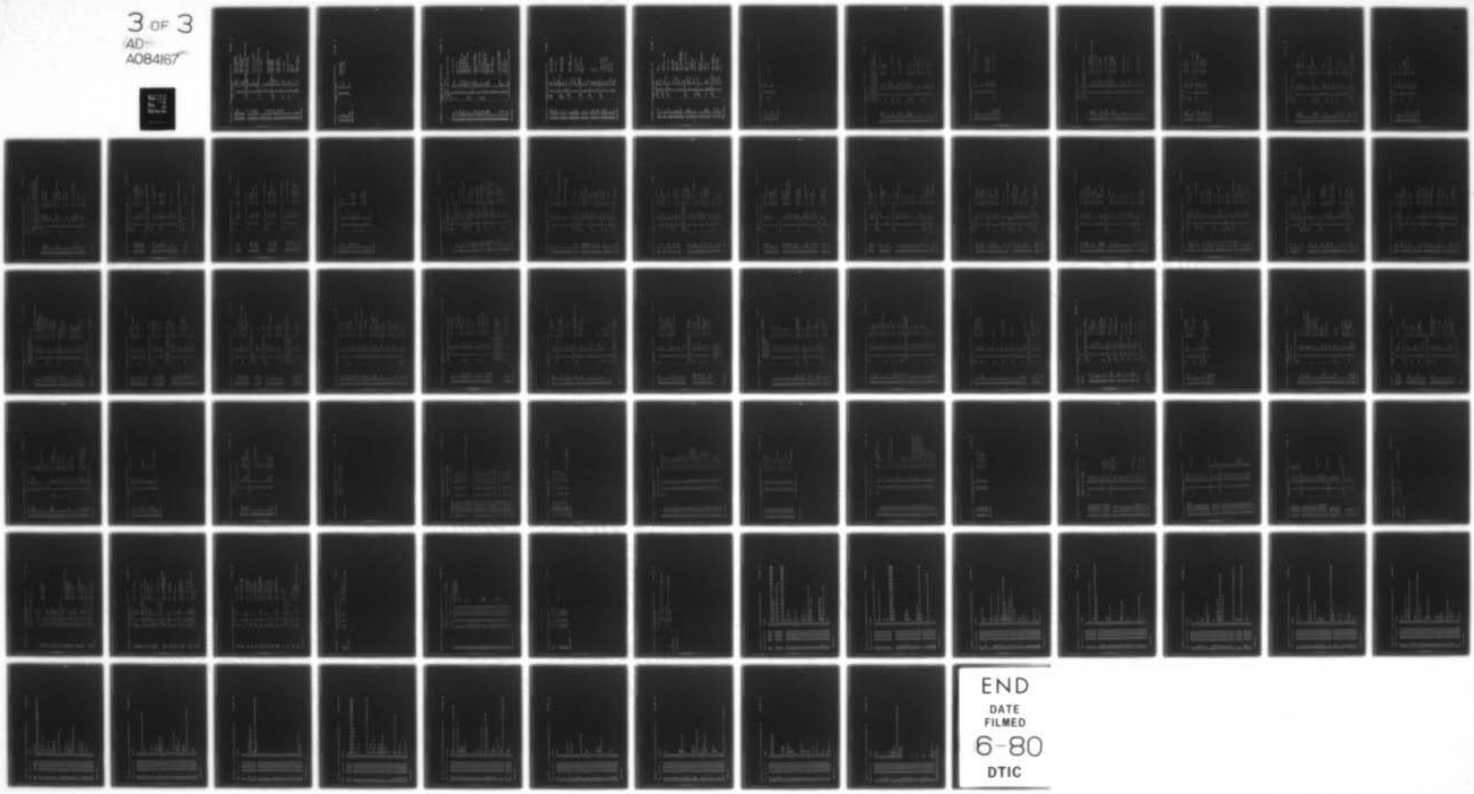
```
                        :**** COMMAND -- LINE NUMBER
                        :*
                        :* THIS ROUTINE IS USED TO ENTER LINES INTO THE FILE
                        :*  AREA.  THE LINE NUMBER IS FIRST CHECKED TO SEE IF IT IS
                        :*  A VALID NUMBER (0000-9999).  NEXT IT IS CHECKED TO SEE
                        :*  IF IT IS GREATER THAN THE MAXIUM CURRENT LINE NUMBER.
                        :*  IF IT IS, THE NEXT LINE IS INSERTED AT THE END OF THE
                        :*  CURRENT FILE AND THE MAXIMUM LINE NUMBER IS UPDATED AS
                        :*  WELL AS THE END OF FILE POSITION.  LINE NUMBERS THAT
                        :*  ALREADY EXIST ARE INSERTED INTO THE FILE AREA AT THE
                        :*  APPROPRIATE PLACE AND ANY EXTRA CHARACTERS IN THE OLD
                        :*  LINE ARE DELETED.
                        :*
0F76                    :LINE   EQU     $
0F76  CD CD02           :       CALL    FECHK           ; PRIMARY FILE MUST EXIST
0F79  0E04              :       LD      C,4             ; NO OF DIGITS TO CHECK
0F7B  21 00FE           :       LD      HL,IBUF-1       ; INITIALIZE ADDRESS
0F7E  35                :       DEC     (HL)            ; CORRECT CHAR COUNT IN LINE
0F7F  23                :LICK   INC     HL
0F80  7E                :       LD      A,(HL)          ; FETCH LINE DIGIT
0F81  FE30              :       CP      '0'             ; CHECK FOR VALID NUMBER
0F83  DA FF0F           :       JP      C,LFIX
0F86  FE3A              :       CP      '9'+1
0F88  D2 FF0F           :       JP      NC,LFIX
0F8B  0D                :       DEC     C
0F8C  20F1              :       JR      NZ,LICK-$
0F8E  22 E5F1           :LFIXR  LD      (ADDS),HL       ; FIND ADDRESS
0F91  11 9FF0           :       LD      DE,MAXL+3       ; GET ADDRESS
0F94  CD 1503           :       CALL    COMO
0F97  3018              :       JR      NC,INSR-$
                        :*  GET HERE IF NEW LINE IS GREATER THAN MAXIMUM LINE NO
0F99  23                :       INC     HL
0F9A  CD 0503           :       CALL    LODM            ; GET NEW LINE NUMBER
0F9D  21 9FF0           :       LD      HL,MAXL+3
0FA0  CD 0D03           :       CALL    STOM            ; MAKE IT MAXIMUM LINE NUMBER
0FA3  11 00FE           :       LD      DE,IBUF-1
0FA6  2A 9AF0           :       LD      HL,(EOFP)       ; END OF FILE POSITION
0FA9  0E01              :       LD      C,1
0FAB  CD 7ED0           :       CALL    LMOV            ; PLACE LINE IN FILE
0FAE  3601              :       LD      (HL),1          ; END OF FILE INDICATOR
0FB0  22 9AF0           :       LD      (EOFP),HL       ; END OF FILE ADDRESS
0FB3  C9                :       RET
                        :* GET HERE IF NEW LINE MUST BE INSERTED INTO ALREADY EXISTING FILE AREA
0FB4  CD DE02           :INSR   CALL    FIND1           ; FIND LINE IN FILE
0FB7  0E02              :       LD      C,2
0FB9  2801              :       JR      Z,EQUL-$
0FBB  0D                :       DEC     C               ; NEW LINE NOT EQUAL TO SOME OLD LINE
0FBC  46                :EQUL   LD      B,(HL)
0FBD  28                :       DEC     HL
```

PROGRAM ERRORS -- 0

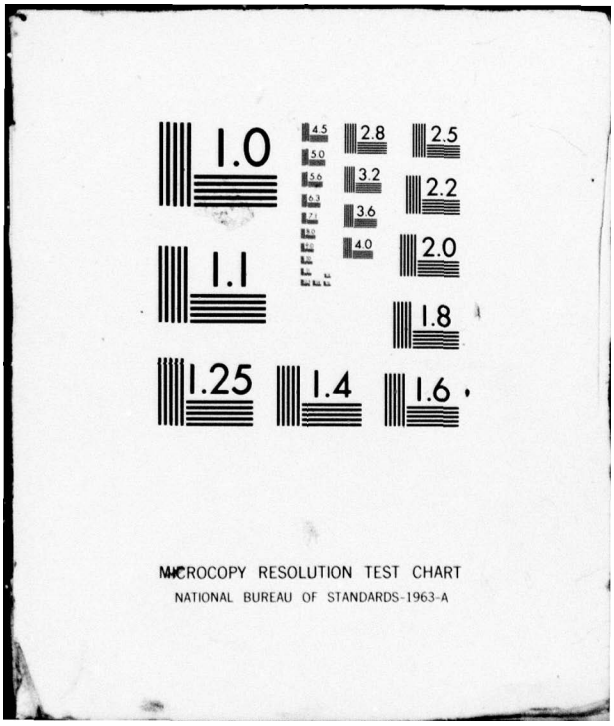
AD-A084 167 ARMY SATELLITE COMMUNICATIONS AGENCY FORT MONMOUTH NJ F/G 9/2
PROJECT ARIES. USER'S MANUALS FOR ARIAN II AND ASSOCIATED SUBSY--ETC(U)
APR 80 R L CONN

UNCLASSIFIED NL

3 OF 3
AD-
A084167



END
DATE
FILMED
6-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
0FBE  3602              :        LD      (HL),2          ; MOVE LINE INDICATOR
0FC0  22 E3F1           :        LD      (INSP),HL       ; INSERT LINE POSITION
0FC3  3A 00FE           :        LD      A,(IBUF-1)      ; NEW LINE COUNT
0FC6  0D                :        DEC     C
0FC7  2805              :        JR      Z,LT-$          ; NEW LINE NOT = OLD LINE
0FC9  90                :        SUB     B               ; COUNT DIFFERENCE
0FCA  2824              :        JR      Z,ZERO-$        ; LINE LENGTHS EQUAL
0FCC  3812              :        JR      C,GT-$
                        :*  GET HERE IF NO OF CHARS IN OLD LINE > NO OF CHARS IN NEW LINE OR NEW
                        :*    LINE NO WAS NOT EQUAL TO SOME OLD LINE NO
0FCE  2A 9AF0           :LT      LD      HL,(EOFP)       ; END OF FILE ADDRESS
0FD1  54                :        LD      D,H
0FD2  5D                :        LD      E,L
0FD3  CD 11D1           :        CALL    ADR
0FD6  22 9AF0           :        LD      (EOFP),HL       ; NEW END OF FILE ADDRESS
0FD9  0E02              :        LD      C,2
0FDB  CD 87D0           :        CALL    RMOV            ; OPEN UP FILE AREA
0FDE  1810              :        JR      ZERO-$
                        :*  GET HERE IF NO OF CHARS IN OLD LINE < NO OF CHARS IN NEW LINE.
0FE0  2F                :GT      CPL
0FE1  3C                :        INC     A               ; COUNT DIFFERENCE
0FE2  54                :        LD      D,H
0FE3  5D                :        LD      E,L
0FE4  CD 11D1           :        CALL    ADR
0FE7  EB                :        EX      DE,HL
0FE8  CD 7ED0           :        CALL    LMOV            ; DELETE EXCESS CHAR IN FILE
0FEB  3601              :        LD      (HL),1          ; E-O-F INDICATOR
0FED  22 9AF0           :        LD      (EOFP),HL       ; E-O-F ADDRESS
                        :*  GET HERE TO INSERT CURRENT LINE INTO FILE AREA
0FF0  2A E3F1           :ZERO    LD      HL,(INSP)       ; INSERT ADDRESS
0FF3  360D              :        LD      (HL),0DH
0FF5  23                :        INC     HL
0FF6  11 00FE           :        LD      DE,IBUF-1       ; NEW LINE ADDRESS
0FF9  0E01              :        LD      C,1             ; CHECK VALUE
0FFB  CD 7ED0           :        CALL    LMOV            ; PLACE LINE IN FILE
0FFE  C9                :        RET
0FFF  21 00FE           :LFIX    LD      HL,IBUF-1
1002  46                :        LD      B,(HL)          ; GET NO CHARS
1003  58                :        LD      E,B
1004  1600              :        LD      D,0
1006  19                :        ADD     HL,DE           ; PT TO LAST CHAR IN LINE
1007  7E                :LFIXL   LD      A,(HL)          ; MOVE LINE
1008  23                :        INC     HL
1009  77                :        LD      (HL),A          ; DOWN ONE
100A  2B                :        DEC     HL              ; BACK UP
100B  2B                :        DEC     HL
100C  10F9              :        DJNZ    LFIXL-$         ; DONE?
100E  23                :        INC     HL              ; PT TO 1ST CHAR
100F  3630              :        LD      (HL),'0'        ; INSERT AN ASCII ZERO

PROGRAM ERRORS --      0
```

```
1011  2B              :       DEC     HL
1012  34              :       INC     (HL)            ; INCR NO CHARS
1013  0D              :       DEC     C               ; INSERTIONS DONE?
1014  20E9            :       JR      NZ,LFIX-$
1016  23              :       INC     HL              ; PT TO LAST DIGIT
1017  23              :       INC     HL
1018  23              :       INC     HL
1019  23              :       INC     HL
101A  C3 8E0F         :       JP      LFIXR
```

PROGRAM ERRORS -- 0

```
                        :**** COMMAND -- LIST, PRINT
                        :*
                        :* THIS ROUTINE IS USED TO LIST THE CONTENTS OF THE FILE
                        :*    AREA
                        :*
101D                    :LISTP  EQU     $
101D  F680              :       OR      80H             ; MASK IN PRINT FLAG
101F  32 F9F1           :       LD      (SPCHAR),A
1022                    :LIST   EQU     $
1022  CD A903           :       CALL    LINIT           ; SET LINE COUNT
1025  AF                :       XOR     A               ; A=0
1026  32 14F0           :       LD      (LCTRL),A       ; ASSUME ONE LINE
1029  CD AF02           :       CALL    CKA11           ; CHECK FOR 1ST ARG
102C  2807              :       JR      Z,LSTALL-$      ; LIST ALL OF FILE IF NONE
102E  CD BE02           :       CALL    CKA21           ; CHECK FOR 2ND ARG
1031  2017              :       JR      NZ,LST0-$       ; PRINT GROUP OF LINES
1033  181A              :       JR      LST1-$          ; PRINT ONE LINE
1035  21 FDF1           :LSTALL LD      HL,ABUF         ; STORE '0000' AS FIRST LINE
1038  0604              :       LD      B,4             ; 4 CHARS
103A  3630              :LSTA1  LD      (HL),30H        ; STORE '0'
103C  23                :       INC     HL
103D  10FB              :       DJNZ    LSTA1-$
103F  11 9CF0           :       LD      DE,MAXL         ; STORE MAX LINE NUMBER
1042  21 01F2           :       LD      HL,ABUF+4       ; STORE ENDING LINE NUMBER
1045  0E04              :       LD      C,4             ; 4 CHARS
1047  CD 81D0           :       CALL    LMOVC
104A  3E01              :LST0   LD      A,1             ; LCTRL=1 MEANS UPPER BOUND
104C  32 14F0           :       LD      (LCTRL),A
104F  CD D502           :LST1   CALL    FIND            ; FIND 1ST LINE
1052  23                :LIST0  INC     HL              ; PT TO 1ST DIGIT OF LINE NO
1053  CD 7410           :       CALL    SCRNOC          ; PRINT LINE TO <CR>
1056  CD FD02           :       CALL    EOF             ; CHECK FOR EOF AFTER <CR>
1059  CD 9603           :       CALL    LINECNT         ; LINE PAGING
105C  3A 14F0           :       LD      A,(LCTRL)       ; UPPER BOUND
105F  B7                :       OR      A
1060  C8                :       RET     Z               ; ONE LINE DONE
1061  E5                :       PUSH    HL              ; YES -- ARE WE THERE?
1062  23                :       INC     HL
1063  23                :       INC     HL
1064  23                :       INC     HL
1065  23                :       INC     HL              ; PT TO LAST DIGIT OF LINE NO
1066  11 04F2           :       LD      DE,ABUF+7       ; PT TO LAST DIGIT OF LIMIT
1069  EB                :       EX      DE,HL
106A  CD 2603           :       CALL    COM1            ; BEYOND LIMIT?
106D  E1                :       POP     HL
106E  D0                :       RET     NC              ; CARRY=0 MEANS YES
106F  CD F400           :       CALL    INK             ; CHECK FOR <ESC>
1072  18DE              :       JR      LIST0-$
                        :*  THIS ROUTINE OUTPUTS A LINE TO PRINCIPAL I/O CHAN OR PRINTER

PROGRAM ERRORS --    0
```

```
1074  CD A710           :SCRNOC  CALL    CRLFP           ; OUTPUT <CR> <LF>
1077  CD 9003           :SCRNO   CALL    GETSP           ; GET SPCHAR & MASK
107A  FE4E              :        CP      'N'
107C  2006              :        JR      NZ,OUTLN-$
107E  01 0500           :        LD      BC,5            ; SKIP 5 CHARS
1081  09                :        ADD     HL,BC
1082  1809              :        JR      OUTLNE-$
                        :*  OUTPUT LINE NUMBER AND LINE
1084  0605              :OUTLN   LD      B,5             ; COPY NUMBER & <SP>
1086  7E                :OUTLN1  LD      A,(HL)          ; GET AND PRINT CHAR
1087  CD 0B11           :        CALL    OUTAP
108A  23                :        INC     HL
108B  10F9              :        DJNZ    OUTLN1-$
                        :*  OUTPUT LINE
108D  7E                :OUTLNE  LD      A,(HL)          ; GET 1ST CHAR
108E  FE3B              :        CP      ';'             ; PRINT COMMENTS DIRECTLY.
1090  280B              :        JR      Z,OUTLN2-$
1092  FE2A              :        CP      '*'
1094  2807              :        JR      Z,OUTLN2-$
1096  CD 9003           :        CALL    GETSP           ; GET SPCHAR & MASK
1099  FE46              :        CP      'F'
109B  281C              :        JR      Z,OUTLN3-$      ; FORMAT LINE
                        :*  OUTPUT LINE UNFORMATTED
109D  7E                :OUTLN2  LD      A,(HL)          ; GET AND PRINT CHAR
109E  FE0D              :        CP      0DH             ; DONE?
10A0  C8                :        RET     Z
10A1  CD 0B11           :        CALL    OUTAP
10A4  23                :        INC     HL
10A5  18F6              :        JR      OUTLN2-$
                        :*  OUTPUT CRLF
10A7  3E0A              :CRLFP   LD      A,0AH           ; <LF>
10A9  CD 0B11           :        CALL    OUTAP
10AC  3E0D              :        LD      A,0DH           ; <CR>
10AE  CD 0B11           :        CALL    OUTAP
10B1  3E7F              :        LD      A,7FH           ; <DEL>
10B3  CD 0B11           :        CALL    OUTAP
10B6  C3 0B11           :        JP      OUTAP
                        :*  OUTPUT LINE FORMATTED
10B9  0607              :OUTLN3  LD      B,7             ; 7 CHARS IN FIELD 1
10BB  CD D610           :        CALL    OUTLNB
10BE  0605              :        LD      B,5             ; 5 CHARS IN FIELD 2
10C0  CD D610           :        CALL    OUTLNB
10C3  0609              :        LD      B,9             ; 8 CHARS IN FIELD 3
10C5  7E                :        LD      A,(HL)
10C6  FE3B              :        CP      ';'             ; NO FIELD 3?
10C8  2806              :        JR      Z,OUTLN4-$
10CA  05                :        DEC     B               ; ADJUST CNT
10CB  CD D610           :        CALL    OUTLNB
10CE  18CD              :        JR      OUTLN2-$

PROGRAM ERRORS --     0
```

```
                        :*  OUTPUT END COMMENT
10D0  CD FE10           :OUTLN4  CALL    OUTLB1          ; TAB
10D3  2B                :        DEC     HL              ; PT TO ';'
10D4  18C7              :        JR      OUTLN2-$
                        :*  OUTPUT FIELD
10D6  7E                :OUTLNB  LD      A,(HL)          ; GET CHAR
10D7  FE27              :        CP      27H             ; SINGLE QOUTE -- STRING
10D9  2014              :        JR      NZ,OUTLNC-$
                        :*  OUTPUT STRING
10DB  CD 0B11           :OUTQ    CALL    OUTAP           ; HANDLE STRING
10DE  23                :        INC     HL
10DF  05                :        DEC     B
10E0  7E                :        LD      A,(HL)          ; GET NEXT CHAR
10E1  FE20              :        CP      ' '             ; INVALID EOL?
10E3  3817              :        JR      C,OUTLB0-$
10E5  FE27              :        CP      27H             ; STRING END?
10E7  20F2              :        JR      NZ,OUTQ-$       ; CONTINUE
10E9  05                :        DEC     B
10EA  CD 0B11           :        CALL    OUTAP           ; PRINT END SINGLE QUOTE
10ED  23                :        INC     HL              ; PT TO NEXT CHAR AFTER STR
10EE  7E                :        LD      A,(HL)          ; GET IT
                        :*  OUTPUT SIMPLE CHARS
10EF  FE20              :OUTLNC  CP      ' '             ; CHECK FOR <CR> OR <SP>
10F1  3809              :        JR      C,OUTLB0-$      ; <CR>
10F3  CD 0B11           :        CALL    OUTAP           ; PRINT CHAR
10F6  2806              :        JR      Z,OUTLB1-$
10F8  23                :        INC     HL              ; PT TO NEXT
10F9  05                :        DEC     B               ; DECREMENT COUNT
10FA  18DA              :        JR      OUTLNB-$
10FC  C1                :OUTLB0  POP     BC              ; CLEAR STACK
10FD  C9                :        RET
                        :*  SPACE FILL LINE FIELD
10FE  78                :OUTLB1  LD      A,B             ; GET COUNT
10FF  B7                :        OR      A               ; SET FLAGS
1100  F8                :        RET     M               ; RETURN IF BEYOND LIMIT
1101  C8                :        RET     Z
1102  3E20              :        LD      A,' '           ; <SP> FILL
1104  CD 0B11           :OUTLB2  CALL    OUTAP           ; PRINT <SP>
1107  10FB              :        DJNZ    OUTLB2-$
1109  23                :        INC     HL              ; PT TO NEXT
110A  C9                :        RET
                        :*  LIST/PRINT OUTPUT ROUTINE
110B  F5                :OUTAP   PUSH    AF              ; SAVE A AND FLAGS
110C  3A F9F1           :        LD      A,(SPCHAR)      ; CHECK FOR PRINT FLAG
110F  B7                :        OR      A
1110  FA 1E11           :        JP      M,OUTAP3        ; PRINT IF SET
1113  FE50              :        CP      'P'             ; EXPLICIT PRINT?
1115  2807              :        JR      Z,OUTAP3-$
1117  F1                :        POP     AF              ; GET A
```

PROGRAM ERRORS -- 0

```
1118  F5                :         PUSH    AF
1119  CD 1ED0           :         CALL    OUTPUT
111C  F1                :         POP     AF
111D  C9                :         RET
111E  F1                :OUTAP3   POP     AF              ; GET CHAR
111F  F5                :         PUSH    AF
1120  CD 36D0           :         CALL    OUT3A
1123  F1                :         POP     AF
1124  C9                :         RET

PROGRAM ERRORS --      0
```

```
                        :**** COMMAND -- RNUM
                        :*
                        :*  RNUM -- RENUMBER PRIMARY FILE: THIS CMND RENUMBERS
                        :*  THE PRIMARY FILE.  FORMAT   RNUM (D4) (D4)
                        :*  THE FIRST D4 IS THE OPTIONAL STARTING LINE NUMBER, THE
                        :*  SECOND D4 IS THE OPTIONAL INCREMENT.  IF NO ARGS ARE PRESENT,
                        :*  THE VALUE OF 10 (DECIMAL) WILL BE DEFAULT.
                        :*
1125                    :RNUM    EQU     $
1125  CD CD02           :        CALL    FECHK           ; PRIMARY FILE MUST EXIST
1128  CD 2E11           :        CALL    RNUM0           ; RENUMBER
112B  C3 D60B           :        JP      FFIX            ; FIX EOFP & MAXL
112E  2A FDF1           :RNUM0   LD      HL,(ABUF)       ; CHECK PARAMS
1131  AF                :        XOR     A               ; 1ST PARAM = 0?
1132  BD                :        CP      L
1133  200A              :        JR      NZ,RNUM1-$
1135  BC                :        CP      H
1136  2007              :        JR      NZ,RNUM1-$
1138  21 0500           :RENME   LD      HL,05H          ; SET START LINE & INCR TO 5
113B  5D                :        LD      E,L             ; D&E H&L
113C  54                :        LD      D,H
113D  1810              :        JR      RNUM2-$
113F  2A 0DF2           :RNUM1   LD      HL,(BBUF)
1142  EB                :        EX      DE,HL           ; D&E STARTING LINE NO
1143  2A 0FF2           :        LD      HL,(BBUF+2)     ; 2ND PARAM =0?
1146  BD                :        CP      L
1147  2006              :        JR      NZ,RNUM2-$
1149  BC                :        CP      H
114A  2003              :        JR      NZ,RNUM2-$
114C  21 0500           :        LD      HL,05H          ; SET INCR TO 5
114F  22 16F0           :RNUM2   LD      (LINC),HL       ; SAVE INCR
1152  2A 98F0           :        LD      HL,(BOF%)       ; H&L PT TO 1ST LINE
1155  7E                :RNUML   LD      A,(HL)          ; GET 1ST CHAR
1156  FE01              :        CP      1               ; EOF?
1158  C8                :        RET     Z               ; DONE IF SO
1159  E5                :        PUSH    HL              ; SAVE PTR & NEXT LINE NO
115A  D5                :        PUSH    DE
115B  0E04              :        LD      C,4             ; STORE NEW LINE NO
115D  EB                :        EX      DE,HL           ; H&L = NEXT LINE NO
115E  0604              :GNO     LD      B,4             ; 4 BITS/DIGIT
1160  AF                :        XOR     A               ; A=0
1161  29                :GNO1    ADD     HL,HL           ; A=HIGH NYBBLE OF H&L
1162  17                :        RLA
1163  10FC              :        DJNZ    GNO1-$
1165  C630              :        ADD     A,30H           ; CONVERT TO ASCII
1167  13                :        INC     DE              ; STORE CHAR
1168  12                :        LD      (DE),A
1169  0D                :        DEC     C               ; 4 CHARS STORED?
116A  20F2              :        JR      NZ,GNO-$

PROGRAM ERRORS --    0
```

```
116C  D1              :        POP     DE              ; COMPUTE NEW LINE NO
116D  2A 16F0         :        LD      HL,(LINC)
1170  7D              :        LD      A,L             ; COMPUTE BCD SUM OF D&E AND H&L
1171  83              :        ADD     A,E
1172  27              :        DAA
1173  5F              :        LD      E,A
1174  7C              :        LD      A,H
1175  8A              :        ADC     A,D
1176  27              :        DAA
1177  57              :        LD      D,A             ; D&E NOW NEW LINE NO
1178  3807            :        JR      C,RNUME-$       ; ERROR IF > 9999
117A  E1              :        POP     HL              ; ADVANCE TO NEXT LINE
117B  7E              :        LD      A,(HL)          ; CHAR COUNT IN A
117C  CD 11D1         :        CALL    ADR
117F  18D4            :        JR      RNUML-$
1181  21 471C         :RNUME   LD      HL,MSS0         ; 'LINE NUMBER OVERFLOW'
1184  C3 6603         :        JP      MESS

PROGRAM ERRORS --      0
```

```
                        :**** COMMAND -- DEL
                        :*
                        :* THIS ROUTINE IS USED TO DELETE LINES FROM THE
                        :*  FILE AREA.  THE REMAINING FILE AREA IS THEN MOVED IN
                        :*  MEMORY SO THAT THERE IS NO EXCESS SPACE.
                        :*
1187                    :DELL   EQU     $
1187  CD A502           :       CALL    CKA1            ; CHECK FOR PARAMETER
118A  CD D502           :       CALL    FIND            ; FIND LINE IN FILE AREA
118D  DA 390A           :       JP      C,RNERR         ; LINE NOT MATCHED EXACTLY; DELETION ERROR
1190  22 E3F1           :       LD      (DELP),HL       ; SAVE DELETE POSITION
1193  21 04F2           :       LD      HL,ABUF+7
1196  7E                :       LD      A,(HL)          ; CHECK FOR 2ND PARAMETER
1197  B7                :       OR      A               ; SET FLAGS
1198  2003              :       JR      NZ,DEL1-$
119A  21 00F2           :       LD      HL,ABUF+3       ; USE FIRST PARAMETER
119D  22 E5F1           :DEL1   LD      (ADDS),HL       ; SAVE FIND ADDRESS
11A0  EB                :       EX      DE,HL
11A1  21 9FF0           :       LD      HL,MAXL+3
11A4  CD 1503           :       CALL    COMO            ; COMPARE LINE NUMBERS
11A7  2A E3F1           :       LD      HL,(DELP)       ; LOAD DELETE POSITION
11AA  3837              :       JR      C,NOVR-$
                        :*  GET HERE IF DELETION INVOLVES END OF FILE
11AC  22 9AF0           :       LD      (EOFP),HL       ; CHANGE E-O-F POSITION
11AF  3601              :       LD      (HL),1          ; SET E-O-F INDICATOR
11B1  ED5B 98F0         :       LD      DE,(BOFP)       ; GET BEGIN FILE ADDRESS
11B5  060D              :       LD      B,13            ; SET SCAN SWITCH
11B7  2B                :       DEC     HL              ; CHECK FOR BOF
11B8  7D                :DEL2   LD      A,L
11B9  93                :       SUB     E
11BA  7C                :       LD      A,H
11BB  9A                :       SBC     A,D
11BC  3E0D              :       LD      A,0DH           ; LOOK FOR CR
11BE  381B              :       JR      C,DEL4-$        ; DECREMENTED PAST BOF
11C0  05                :       DEC     B
11C1  2B                :       DEC     HL
11C2  BE                :       CP      (HL)            ; FIND NEW MAX LINE
11C3  20F3              :       JR      NZ,DEL2-$
11C5  2B                :       DEC     HL
11C6  7D                :       LD      A,L
11C7  93                :       SUB     E
11C8  7C                :       LD      A,H
11C9  9A                :       SBC     A,D
11CA  3810              :       JR      C,DEL5-$
11CC  BE                :       CP      (HL)            ; END OF PREVIOUS LINE
11CD  23                :       INC     HL
11CE  23                :       INC     HL
11CF  2801              :       JR      Z,DEL3-$
11D1  23                :       INC     HL
```

```
11D2  CD 0503           :DEL3   CALL    LODM            ; LOAD NEW MAX LINE
11D5  21 9FF0           :       LD      HL,MAXL+3       ; SET ADDRESS
11D8  C3 0D03           :       JP      STOM            ; STORE NEW MAX LINE
11DB  B8                :DEL4   CP      B               ; CHECK SWITCH
11DC  EB                :DEL5   EX      DE,HL
11DD  20F2              :       JR      NZ,DEL3-1-$
11DF  32 9CF0           :       LD      (MAXL),A        ; MAKE MAX LINE A SMALL NUMBER
11E2  C9                :       RET
                        :* GET HERE IF DELETION IS IN MIDDLE OF FILE AREA
11E3  CD E602           :NOVR   CALL    FIND2           ; FIND END OF DELETE AREA
11E6  CC E702           :       CALL    Z,FIND3         ; NEXT LINE IF THIS LINE EQUAL
11E9  EB                :       EX      DE,HL
11EA  2A E3F1           :       LD      HL,(DELP)       ; CHAR MOVE TO POSITION
11ED  0E01              :       LD      C,1             ; MOVE TERMINATOR
11EF  CD 7ED0           :       CALL    LMOV            ; COMPACT FILE AREA
11F2  22 9AF0           :       LD      (EOFP),HL       ; SET EOF POSITION
11F5  3601              :       LD      (HL),1          ; SET EOF INDICATOR
11F7  C9                :       RET
```

PROGRAM ERRORS -- 0

```
                        :**** COMMAND -- SYMT
                        :*
                        :*  SYMT -- PRINT THE SYMBOL TABLE OF AN ASSEMBLY
                        :*
11F8                    :SYMB   EQU     $
11F8  CD A903           :       CALL    LINIT           ; SET LINE COUNT
11FB  CD 9F02           :       CALL    CKEN1           ; SYMBOL SPECIFIED?
11FE  C2 4512           :       JP      NZ,SCAN         ; SCAN FOR SYMBOL IF SO
1201  ED5B 75F2         :       LD      DE,(NOLA)       ; GET SYMBOL COUNT
1205  7B                :       LD      A,E             ; ANY SYMBOLS?
1206  D619              :       SUB     NSYM            ; SUBTRACT COUNT OF SYS SYMBOLS
1208  5F                :       LD      E,A
1209  7A                :       LD      A,D
120A  DE00              :       SBC     A,0             ; CHECK CARRY
120C  57                :       LD      D,A
120D  B3                :       OR      E               ; ANY SYMBOLS?
120E  C8                :       RET     Z               ; DONE IF ZERO
120F  21 C8F3           :       LD      HL,SYMT+NSYM*8  ; POINT TO USER SYMBOL TABLE
1212  0604              :SYMB1  LD      B,4             ; 4 SYMBOLS PER LINE
1214  CD 3FD0           :       CALL    CRLF
1217  CD 2512           :SYMB2  CALL    SYMBP           ; PRINT ENTRY
121A  1B                :       DEC     DE              ; DECR COUNT
121B  7A                :       LD      A,D
121C  B3                :       OR      E
121D  C8                :       RET     Z               ; DONE IF ZERO
121E  10F7              :       DJNZ    SYMB2-$
1220  CD 9603           :       CALL    LINECNT         ; CHECK FOR PAGING
1223  18ED              :       JR      SYMB1-$
1225  C5                :SYMBP  PUSH    BC              ; SAVE B&C
1226  D5                :       PUSH    DE              ; SAVE D&E
1227  0606              :       LD      B,LLAB          ; LLAB CHARS PER SYMBOL
1229  7E                :SYMP1  LD      A,(HL)          ; GET CHAR
122A  B7                :       OR      A               ; <SP> IF 0
122B  2002              :       JR      NZ,SYMP2-$
122D  3E20              :       LD      A,' '
122F  CD 1ED0           :SYMP2  CALL    OUTPUT          ; PRINT CHAR
1232  23                :       INC     HL              ; INCR MEMORY PTR
1233  10F4              :       DJNZ    SYMP1-$
1235  CD 5DD0           :       CALL    PCHAR           ; RELATIONAL OPERATOR
1238  3D                :       DB      '='
1239  CD 5003           :       CALL    PDEADR          ; PRINT ADR
123C  23                :       INC     HL              ; PT TO NEXT SYMBOL
123D  D1                :       POP     DE              ; RESTORE DE
123E  C1                :       POP     BC              ; RESTORE BC
123F  CD 42D0           :       CALL    BLK1
1242  C3 42D0           :       JP      BLK1
                        :*  THIS ROUTINE SCANS THE SYMBOL TABLE FOR THE SPECIFIED SYMBOL
1245  0606              :SCAN   LD      B,LLAB          ; CHANGE <SP> TO 0 IN FBUF
1247  21 E9F1           :       LD      HL,FBUF         ; PT TO SYMBOL TO SEARCH FOR
```

```
124A  E5              :         PUSH    HL              ; SAVE PTR TO FBUF
124B  7E              :SCANC1   LD      A,(HL)          ; GET CHAR
124C  FE20            :         CP      ' '             ; <SP>?
124E  2002            :         JR      NZ,SCANC2-$
1250  3600            :         LD      (HL),0          ; FILL WITH <NULL>S
1252  23              :SCANC2   INC     HL              ; PT TO NEXT
1253  10F6            :         DJNZ    SCANC1-$
1255  E1              :         POP     HL              ; RESTORE PTR TO FBUF
1256  ED4B 75F2       :         LD      BC,(NOLA)       ; NUMBER OF SYMBOLS
125A  CD 6818         :         CALL    SCANL           ; SCAN FOR SYMBOL
125D  C0              :         RET     NZ              ; NOT FOUND
125E  0608            :         LD      B,LLAB+2        ; BACK UP TO START OF ENTRY
1260  1B              :SCANL1   DEC     DE
1261  10FD            :         DJNZ    SCANL1-$
1263  CD 3FD0         :         CALL    CRLF            ; NEW LINE
1266  EB              :         EX      DE,HL           ; HL PTS TO ENTRY
1267  C3 2512         :         JP      SYMBP

PROGRAM ERRORS --      0
```

```
                        ;********                                  ********
                        ;******** BINARY FILE DIRECTORY ROUTINES ********
                        ;********                                  ********
                        ;*
                        ;*  BFSCAN -- SCAN BINARY FILE DIRECTORY FOR AN ENTRY WHOSE NAME IS
                        ;*     IN FBUF
                        ;*  IF FOUND, HL POINT TO THE FIRST BYTE OF THE NAME ENTRY AND ZERO
                        ;*     IS OFF; IF NOT FOUND AND DIRECTORY FULL, ZERO IS ON; IF NOT
                        ;*     FOUND AND DIRECTORY NOT FULL, ZERO IS OFF AND HL POINT TO FIRST BYTE
                        ;*     OF NEXT AVAILABLE ENTRY SPACE
                        ;*
126A                    BFSCAN  EQU     $
126A  060A              :       LD      B,MAXBFIL       ; NUMBER OF BINARY FILES
126C  21 18F0           :       LD      HL,BFDIR        ; POINT TO DIRECTORY
126F  0E08              BFS1    LD      C,NMLEN         ; NAME LENGTH
1271  11 E9F1           :       LD      DE,FBUF         ; POINT TO NAME IN FBUF
1274  1A                BFS2    LD      A,(DE)          ; GET CHAR
1275  BE                :       CP      (HL)            ; COMPARE
1276  200D              :       JR      NZ,BFSF-$       ; FAIL
1278  23                :       INC     HL              ; CHECK NEXT
1279  13                :       INC     DE
127A  0D                :       DEC     C               ; DECREMENT COUNT
127B  20F7              :       JR      NZ,BFS2-$       ; CONTINUE IF NOT DONE
127D  11 F8FF           :       LD      DE,0-NMLEN      ; POINT TO BEGINNING OF ENTRY FOUND
1280  19                :       ADD     HL,DE
1281  3E01              BFS3    LD      A,1             ; NOT ZERO MEANS FOUND
1283  B7                :       OR      A
1284  C9                :       RET
1285  23                BFSF    INC     HL              ; SKIP REST OF NAME
1286  0D                :       DEC     C               ; DECREMENT COUNT
1287  20FC              :       JR      NZ,BFSF-$
1289  23                :       INC     HL              ; POINT TO NEXT ENTRY
128A  23                :       INC     HL
128B  23                :       INC     HL
128C  23                :       INC     HL
128D  10E0              :       DJNZ    BFS1-$          ; CONT IF NOT DONE W/DIRECTORY
128F  060A              :       LD      B,MAXFIL        ; NOT FOUND -- ANY EMPTY ENTRIES?
1291  21 18F0           :       LD      HL,BFDIR
1294  11 0C00           :       LD      DE,NMLEN+4
1297  7E                BFS4    LD      A,(HL)          ; GET FIRST BYTE
1298  FE20              :       CP      ' '             ; EMPTY ENTRY IF <SP>
129A  28E5              :       JR      Z,BFS3-$
129C  19                :       ADD     HL,DE           ; SKIP TO NEXT
129D  10F8              :       DJNZ    BFS4-$          ; NEXT
129F  C9                :       RET                     ; ZERO MEANS DIR FULL
                        :
                        :
                        :
                        :*
```

```
                        ;*  FILEB -- CREATE LOCAL BINARY FILE EXPLICITLY
                        ;*
12A0                    ;FILEB   EQU     $
12A0  CD 9502           ;        CALL    CKFN            ; MUST HAVE A FILE NAME
12A3  CD AF02           ;        CALL    CKA11           ; CHECK FOR TWO ARGUMENTS
12A6  CA B812           ;        JP      Z,BFDENT        ; NO ARGUMENT -- USE DEFAULT
12A9  CD B402           ;        CALL    CKA2            ; MUST HAVE TWO ARGUMENTS
12AC  2A 0DF2           ;        LD      HL,(BBUF)       ; GET AND STORE FIRST
12AF  22 6AF2           ;        LD      (ASMST),HL
12B2  2A 0FF2           ;        LD      HL,(BBUF+2)     ; GET AND STORE SECOND
12B5  22 6CF2           ;        LD      (ASMEN),HL
                        ;*
                        ;*  BFDENT -- BINARY FILE DIRECTORY ENTRY; MAKE AN ENTRY INTO
                        ;*    THE BINARY FILE DIRECTORY OF THE FILE POINTED TO BY
                        ;*    (ASMST), (ASMEN) AND NAMED BY FBUF
                        ;*
12B8                    ;BFDENT  EQU     $
12B8  CD 6A12           ;        CALL    BFSCAN          ; SCAN BINARY FILE DIRECTORY
12BB  CA 2605           ;        JP      Z,CURE1         ; DIR FULL IF ZERO
12BE  7E                ;        LD      A,(HL)          ; CHECK IF FOUND
12BF  FE20              ;        CP      ' '
12C1  2803              ;        JR      Z,BFDENT1-$
12C3  CD 4503           ;        CALL    PREPMS          ; REPLACE QUERY
12C6  11 E9E1           ;BFDENT1 LD      DE,FBUF         ; STORE FILE NAME
12C9  0E08              ;        LD      C,NMLEN         ; LENGTH OF NAME
12CB  CD 81D0           ;        CALL    LMOVC           ; STORE IT
12CE  ED5B 6AF2         ;        LD      DE,(ASMST)      ; SAVE START ADDRESS
12D2  73                ;        LD      (HL),E
12D3  23                ;        INC     HL
12D4  72                ;        LD      (HL),D
12D5  23                ;        INC     HL
12D6  ED5B 6CF2         ;        LD      DE,(ASMEN)      ; SAVE END ADDRESS
12DA  73                ;        LD      (HL),E
12DB  23                ;        INC     HL
12DC  72                ;        LD      (HL),D
12DD  C9                ;        RET
                        ;*
                        ;*  CLERA -- STORE BLANKS STARTING IN THE LOCATION POINTED TO BY
                        ;*    HL; NUMBER OF BLANKS IS IN C
                        ;*
12DE                    ;CLERA   EQU     $
12DE  3E20              ;        LD      A,' '           ; STORE <SP>
12E0  C3 FB00           ;        JP      CLRA            ; CLEAR NORMALLY
                        ;
                        ;
                        ;
                        ;*
                        ;*  BSCR -- SCRATCH BINARY FILE COMMAND
                        ;*    THE BINARY FILE DIRECTORY IS CLEARED BY THIS SUBROUTINE.

PROGRAM ERRORS --       0
```

```
                        :*
12E3                    :BSCR   EQU     $
12E3  21 18F0           :       LD      HL,BFDIR        ; POINT TO DIRECTORY
12E6  0E78              :       LD      C,NMLEN*MAXBFIL+4*MAXBFIL  ; NUMBER OF BYTES IN DIR
12E8  18F4              :       JR      CLERA-$         ; DO IT
                        :
                        :
                        :
                        :*
                        :*  BDEL -- DELETE ENTRY IN BINARY FILE DIRECTORY
                        :*
12EA                    :BDEL   EQU     $
12EA  CD 6A12           :       CALL    BFSCAN          ; FIND FILE NAME
12ED  CA 6808           :       JP      Z,DRSDF         ; FILE NOT FOUND
12F0  7E                :       LD      A,(HL)          ; BLANK MEANS NOT FOUND
12F1  FE20              :       CP      ' '
12F3  CA 6808           :       JP      Z,DRSDF         ; FILE NOT FOUND
12F6  0E0C              :       LD      C,NMLEN+4       ; NUMBER OF BYTES TO CLEAR
12F8  18E4              :       JR      CLERA-$         ; DO IT
                        :
                        :
                        :
                        :*
                        :*  BNAME -- RENAME BINARY FILE NAME IN BINARY FILE DIRECTORY
                        :*
12FA                    :BNAME  EQU     $
12FA  CD 6A12           :       CALL    BFSCAN          ; SCAN DIRECTORY
12FD  CA 6808           :       JP      Z,DRSDF         ; FILE NOT FOUND
1300  7E                :       LD      A,(HL)          ; <SP> MEANS NOT FOUND ALSO
1301  FE20              :       CP      ' '
1303  C2 280C           :       JP      NZ,RNMEE        ; RENAME ENTRY POINT
1306  C3 6808           :       JP      DRSDF           ; FILE NOT FOUND
                        :
                        :
                        :
                        :*
                        :*  BDIR -- DISPLAY BINARY FILE DIRECTORY AND LAST ASSEMBLY AREA
                        :*
1309                    :BDIR   EQU     $
1309  CD 3FD0           :       CALL    CRLF
130C  2A 6AF2           :       LD      HL,(ASMST)      ; DISPLAY AREA OF LAST ASSEMBLY
130F  CD 6FD0           :       CALL    PHLB
1312  2A 6CF2           :       LD      HL,(ASMEN)
1315  CD 6CD0           :       CALL    PHL
1318  21 18F0           :       LD      HL,BFDIR        ; POINT TO DIRECTORY
131B  3E0A              :       LD      A,MAXBFIL       ; NUMBER OF ENTRIES TO DISPLAY
131D  32 15F0           :       LD      (LPCNT),A       ; STORE COUNTER
1320  0E08              :BDIRL  LD      C,NMLEN         ; NUMBER OF CHARS PER NAME
1322  7E                :       LD      A,(HL)          ; CHECK FOR EMPTY

PROGRAM ERRORS --       0
```

```
1323  FE20              :        CP      ' '
1325  2821              :        JR      Z,BDIRN-$          ; SKIP TO NEXT
1327  CD 3FD0           :        CALL    CRLF               ; NEW ENTRY
132A  46                :BDIRL1  LD      B,(HL)             ; GET CHAR
132B  CD 24D0           :        CALL    OUTB
132E  23                :        INC     HL                 ; NEXT
132F  0D                :        DEC     C
1330  20F8              :        JR      NZ,BDIRL1-$
1332  CD 42D0           :        CALL    BLK1               ; 2 <SP>
1335  CD 42D0           :        CALL    BLK1
1338  CD 5903           :        CALL    PEDADR             ; PRINT START ADR
133B  CD 5903           :        CALL    PEDADR             ; PRINT END ADR
133E  3A 15F0           :BDIRC   LD      A,(LPCNT)          ; CHECK FOR DONE
1341  3D                :        DEC     A
1342  32 15F0           :        LD      (LPCNT),A
1345  20D9              :        JR      NZ,BDIRL-$
1347  C9                :        RET
1348  59                :BDIRN   LD      E,C                ; SKIP C + 4 BYTES
1349  1600              :        LD      D,0
134B  19                :        ADD     HL,DE              ; NMLEN NOW SKIPPED
134C  23                :        INC     HL                 ; SKIP 4 MORE FOR ADR
134D  23                :        INC     HL
134E  23                :        INC     HL
134F  23                :        INC     HL
1350  18EC              :        JR      BDIRC-$

PROGRAM ERRORS --      0
```

```
                        :**** COMMAND -- ASSM
                        :*
                        :*  ASSM -- ASSEMBLER PROGRAM
                        :*  THIS PROGRAM ASSEMBLES THE PRIMARY FILE
                        :*
1352                    :ASSM   EQU     $
                        :*  SAVE RETURN ADDRESS
1352  E1                :       POP     HL              ; GET RET ADR
1353  22 6FF2           :       LD      (ASMRET),HL     ; SAVE IT
                        :*  SET 'X' (FORMATTED PRINT) OPTION
1356  FE58              :       CP      'X'
1358  2005              :       JR      NZ,ASSM0-$
135A  3EC6              :       LD      A,'F'+80H       ; FORMAT WITH PRINT
135C  32 F9F1           :       LD      (SPCHAR),A
                        :*  INITIALIZE ERROR FLAG
135F  AF                :ASSM0  XOR     A               ; SET ERROR FLAG TO NO ERRORS
1360  32 6EF2           :       LD      (SYSERR),A
                        :*  PRIMARY FILE MUST EXIST
1363  CD CD02           :       CALL    FECHK           ; ABORT IF NOT
                        :*  SET ASSEMBLY START ADDRESS
1366  CD AF02           :       CALL    CKA11           ; CHECK FOR PARAMETER
1369  2007              :       JR      NZ,ASSM1-$
136B  2A 02F0           :       LD      HL,(WSPE)       ; ASSEMBLE AT END OF WS
136E  23                :       INC     HL
136F  22 0DF2           :       LD      (BBUF),HL
                        :*  SET ADDRESS TO PLACE CODE GENERATED BY THE ASSEMBLY
1372  CD BE02           :ASSM1  CALL    CKA21           ; CHECK FOR 2ND PARAMETER
1375  2006              :       JR      NZ,ASM1-$
1377  2A 0DF2           :       LD      HL,(BBUF)       ; FETCH 1ST PARAMETER
137A  22 0FF2           :       LD      (BBUF+2),HL     ; STORE INTO 2ND PARAMETER
                        :*  INITIALIZE ASSEMBLER PARAMETERS
137D  2A 0DF2           :ASM1   LD      HL,(BBUF)       ; SET DEFAULT EXECUTION ADDRESS
1380  22 0BF0           :       LD      (EXADR),HL
1383  3A F9F1           :       LD      A,(SPCHAR)      ; FETCH INPUT CHARACTER
1386  D64C              :       SUB     'L'             ; RESET A IF LISTING REQUESTED
1388  32 63F2           :       LD      (AERR),A        ; SAVE ERROR FLAG
                        :*  INITIALIZE SYMBOL TABLE BY LOADING SYSTEM SYMBOLS
138B  21 601D           :       LD      HL,SSYMT        ; STORE SYSTEM SYMBOL TABLE
138E  11 00F3           :       LD      DE,SYMT
1391  4E                :       LD      C,(HL)          ; C=NO SYMBOLS
1392  0600              :       LD      B,0             ; STORE AND INIT SYMT COUNT
1394  ED43 75F2         :       LD      (NOLA),BC       ; NUMBER OF LABELS
1398  23                :       INC     HL              ; POINT TO FIRST ENTRY
1399  0604              :SMTL   LD      B,4             ; 4 CHARS SYMBOL NAME
139B  7E                :SMTL1  LD      A,(HL)          ; GET CHAR
139C  12                :       LD      (DE),A          ; STORE CHAR
139D  23                :       INC     HL              ; INCR PTRS
139E  13                :       INC     DE
139F  10FA              :       DJNZ    SMTL1-$         ; DECR COUNT

PROGRAM ERRORS --       0
```

```
13A1  0602              :       LD      B,LLAB-4        ; FILL REST WITH <NULL>S
13A3  AF                :       XOR     A               ; A=0
13A4  12                :SMTL2  LD      (DE),A          ; STORE <NULL>
13A5  13                :       INC     DE              ; INCR PTR
13A6  10FC              :       DJNZ    SMTL2-$         ; DECR COUNT
13A8  7E                :       LD      A,(HL)          ; GET LOW ADR
13A9  13                :       INC     DE              ; STORE 2ND (TABLE STORED HI,LOW NOT LOW,HI)
13AA  12                :       LD      (DE),A          ; STORE IT
13AB  23                :       INC     HL              ; INCR PTRS
13AC  1B                :       DEC     DE
13AD  7E                :       LD      A,(HL)          ; GET HIGH ADR
13AE  12                :       LD      (DE),A          ; STORE IT FIRST RATHER THAN SECOND
13AF  23                :       INC     HL              ; INCR PTRS
13B0  13                :       INC     DE
13B1  13                :       INC     DE
13B2  0D                :       DEC     C               ; DECR SYMBOL COUNT
13B3  20E4              :       JR      NZ,SMTL-$
                        :*  PRINT 'PASS 1' MESSAGE
13B5  21 5B1C           :       LD      HL,MSG0         ; POINT TO 'ASM PASS 1'
13B8  CD 8A03           :       CALL    SCRN
                        :*  START PASS 1 OF ASSEMBLY !!!!!!!!!!
13BB  AF                :       XOR     A               ; GET A ZERO
                        :*  PASS INITIALIZATION
13BC  32 71F2           :ASM2   LD      (PAS1),A        ; SET PASS INDICATOR
13BF  2A 0DF2           :       LD      HL,(BBUF)       ; FETCH ORIGIN
13C2  22 68F2           :       LD      (ASPC),HL       ; INITIALIZE PC
13C5  22 6AF2           :       LD      (ASMST),HL      ; STORE START OF PROGRAM
13C8  2A 98F0           :       LD      HL,(BOFP)       ; GET START OF FILE
13CB  22 E3F1           :       LD      (APNT),HL
13CE  CD A903           :       CALL    LINIT           ; SET LINE COUNT
                        :*  START PROCESSING A LINE
13D1  2A E3F1           :ASM3   LD      HL,(APNT)       ; FETCH LINE POINTER
13D4  31 00F3           :       LD      SP,STACK
13D7  7E                :       LD      A,(HL)          ; FETCH CHARACTER
13D8  FE01              :       CP      1               ; END OF FILE
13DA  CA 2A17           :       JP      Z,EPASS         ; JUMP IF END OF FILE
13DD  EB                :       EX      DE,HL
13DE  13                :       INC     DE              ; INCREMENT ADDRESS
                        :*  INITIALIZE OUTPUT BUFFER
13DF  21 ECFD           :       LD      HL,OBUF         ; BLANK START ADDRESS
13E2  3EFC              :       LD      A,IBUF-5        ; BLANK END ADDRESS
13E4  BD                :CLER   CP      L
13E5  2805              :       JR      Z,CLER1-$
13E7  3620              :       LD      (HL),' '        ; BLANK OUT BUFFER
13E9  23                :       INC     HL
13EA  18FB              :       JR      CLER-$
                        :*  PLACE CURRENT LINE INTO OUTPUT BUFFER
13EC  0E0D              :CLER1  LD      C,0DH           ; STOP CHARACTER
13EE  CD 7ED0           :       CALL    LMOV            ; MOVE LINE INTO BUFFER
```

PROGRAM ERRORS -- 0

```
13F1  71                :         LD      (HL),C          ; PLACE CR IN BUFFER
13F2  EB                :         EX      DE,HL
13F3  22 E3F1           :         LD      (APNT),HL       ; SAVE ADDRESS
                        :*  CHECK FOR PASS
13F6  3A 71F2           :         LD      A,(PASI)        ; FETCH PASS INDICATOR
13F9  B7                :         OR      A               ; SET FLAGS
13FA  2008              :         JR      NZ,ASM4-$       ; JUMP IF PASS 2
                        :*  ASSEMBLER PASS 1
13FC  CD 4A14           :         CALL    PASS1
13FF  CD F400           :         CALL    INK             ; CHECK FOR INTERRUPT
1402  18CD              :         JR      ASM3-$
                        :*  ASSEMBLER PASS 2
1404  CD 3F15           :ASM4     CALL    PASS2
1407  CD F400           :         CALL    INK             ; CHECK FOR INTERRUPT
                        :*  OUTPUT LINE IF REQUESTED
140A  21 ECFD           :         LD      HL,OBUF         ; OUTPUT BUFFER ADDRESS
140D  CD 1214           :         CALL    AOUT            ; OUTPUT LINE
1410  18BF              :         JR      ASM3-$
                        :*
                        :* THIS ROUTINE IS USED TO OUTPUT THE LISTING FOR
                        :*  AN ASSEMBLY.  IT CHECKS THE ERROR SWITCH TO SEE IF
                        :*  ALL LINES ARE TO BE PRINTED OR JUST THOSE WITH ERRORS.
                        :*
1412  3A ECFD           :AOUT     LD      A,(OBUF)        ; GET MASTER ERROR FLAG
1415  FE20              :         CP      ' '             ; <SP> MEANS NO ERROR
1417  2805              :         JR      Z,AOUT1-$
1419  3E01              :         LD      A,1             ; SET ERROR FLAG
141B  32 6EF2           :         LD      (SYSERR),A
141E  CD 9003           :AOUT1    CALL    GETSP           ; CHECK FOR EXPLICIT PRINT REQ
1421  FE50              :         CP      'P'             ; PRINT?
1423  2810              :         JR      Z,AOU1-$
1425  FE46              :         CP      'F'             ; FORMAT?
1427  280C              :         JR      Z,AOU1-$
1429  3A 63F2           :         LD      A,(AERR)        ; FETCH ERROR SWITCH
142C  B7                :         OR      A               ; SET FLAGS
142D  2806              :         JR      Z,AOU1-$        ; OUTPUT ALL LINES
142F  3A ECFD           :         LD      A,(OBUF)        ; FETCH ERROR INDICATOR
1432  FE20              :         CP      ' '             ; CHECK FOR AN ERROR
1434  C8                :         RET     Z               ; RETURN IF NO ERROR
1435  21 ECFD           :AOU1     LD      HL,OBUF         ; OUTPUT BUFFER ADDRESS
1438  CD A710           :         CALL    CRLFP
1438  0610              :         LD      B,16            ; 16 CHAR.
143D  7E                :AOU2     LD      A,(HL)          ; GET CHAR
143E  CD 0B11           :         CALL    OUTAP
1441  23                :         INC     HL              ; PT TO NEXT
1442  10F9              :         DJNZ    AOU2-$
1444  CD B410           :         CALL    OUTLN           ; OUTPUT LINE
1447  C3 9603           :         JP      LINECNT         ; PAGE IF DESIRED
                        :*
```

PROGRAM ERRORS -- 0

```
                        ;* PASS 1 OF ASSEMBLER; USED TO FORM SYMBOL TABLE
                        ;*
144A  CD 5B01           ;PASS1   CALL    ZBUF            ; CLEAR BUFFER
144D  32 71F2           :        LD      (PAS1),A        ; SET FOR PASS1
1450  21 01FE           :        LD      HL,IBUF         ; INITIALIZE LINE POINTER
1453  22 73F2           :        LD      (PNTR),HL
1456  7E                :        LD      A,(HL)          ; FETCH CHARACTER
1457  FE20              :        CP      ' '             ; CHECK FOR A BLANK
1459  2840              :        JR      Z,OPC-$         ; JUMP IF NO LABEL
145B  FE2A              :        CP      '*'             ; CHECK FOR COMMENT
145D  C8                :        RET     Z               ; RETURN IF COMMENT
145E  FE3B              :        CP      ';'             ; CHECK FOR ';' COMMENT
1460  C8                :        RET     Z               ; RETURN IF SO
                        ;*
                        ;* PROCESS LABEL
                        ;*
1461  CD 3D18           :        CALL    SLAB            ; GET AND CHECK LABEL
1464  DA FD17           :        JP      C,OPS           ; ERROR IN LABEL
1467  CA 271A           :        JP      Z,ERRD          ; DUPLICATE LABEL
146A  CD B014           :        CALL    LCHK            ; CHECK CHARACTER AFTER LABEL
146D  C2 FD17           :        JP      NZ,OPS          ; ERROR IF NO BLANK
1470  2A 75F2           :        LD      HL,(NOLA)       ; CHECK FOR SYMBOL TABLE OVERFLOW
1473  01 8001           :        LD      BC,MXLAB
1476  AF                :        XOR     A
1477  ED42              :        SBC     HL,BC
1479  D2 2605           :        JP      NC,CUSE1
147C  0606              :        LD      B,LLAB          ; LENGTH OF LABELS
147E  21 FDF1           :        LD      HL,ABUF         ; SET BUFFER ADDRESS
1481  7E                :MLAB    LD      A,(HL)          ; FETCH NEXT CHARACTER
1482  12                :        LD      (DE),A          ; STORE IN SYMBOL TABLE
1483  13                :        INC     DE
1484  23                :        INC     HL
1485  10FA              :        DJNZ    MLAB-$
1487  ED53 66F2         :        LD      (TABA),DE       ; SAVE TABLE ADDRESS FOR EQU
148B  3A 69F2           :        LD      A,(ASPC+1)      ; FETCH PC (HIGH)
148E  12                :        LD      (DE),A
148F  13                :        INC     DE
1490  3A 68F2           :        LD      A,(ASPC)        ; FETCH PC (LOW)
1493  12                :        LD      (DE),A          ; STORE IN TABLE
1494  2A 75F2           :        LD      HL,(NOLA)       ; INCREMENT SYMBOL COUNT
1497  23                :        INC     HL
1498  22 75F2           :        LD      (NOLA),HL
                        ;*
                        ;* PROCESS OPCODE
                        ;*
149B  CD 5B01           ;OPC     CALL    ZBUF            ; ZERO WORKING BUFFER
149E  CD 7903           :        CALL    SBLK            ; SCAN TO OPCODE
14A1  D8                :        RET     C               ; FOUND CARRIAGE RETURN
14A2  CD 8B18           :        CALL    ALPS            ; PLACE OPCODE IN BUFFER

PROGRAM ERRORS --       0
```

```
14A5  FE20            :        CP      ' '             ; CHECK FOR BLANK AFTER OPCODE
14A7  DA 8C17         :        JP      C,OPCD          ; OR AFTER OPCODE
14AA  C2 2418         :        JP      NZ,OERR         ; ERROR IF NO BLANK
14AD  C3 8C17         :        JP      OPCD            ; CHECK OPCODE
                      :*
                      :* THIS ROUTINE CHECKS THE CHARACTER AFTER A LABEL
                      :*  FOR A BLANK OR A COLON.
                      :*
14B0  2A 73F2         :LCHK    LD      HL,(PNTR)
14B3  7E              :        LD      A,(HL)          ; GET CHARACTER AFTER LABEL
14B4  FE20            :        CP      ' '             ; CHECK FOR A BLANK
14B6  C8              :        RET     Z               ; RETURN IF A BLANK
14B7  FE3A            :        CP      3AH             ; CHECK FOR A COLON
14B9  C0              :        RET     NZ
14BA  23              :        INC     HL
14BB  22 73F2         :        LD      (PNTR),HL       ; SAVE POINTER
14BE  C9              :        RET
                      :*
                      :* PROCESS ANY PSEUDO OPS THAT NEED TO BE IN PASS 1
                      :*
14BF  CD 7903         :PSU1    CALL    SBLK            ; SCAN TO OPERAND
14C2  1A              :        LD      A,(DE)          ; FETCH VALUE
14C3  FE14            :        CP      20              ; CHECK FOR Z80 4-BYTE CODES
14C5  3025            :        JR      NC,PSZ81-$
14C7  B7              :        OR      A               ; SET FLAGS
14C8  2828            :        JR      Z,ORG1-$        ; ORG OPCODE
14CA  FE01            :        CP      1
14CC  2838            :        JR      Z,EQU1-$        ; EQU OPCODE
14CE  FE02            :        CP      2               ; DB OPCODE
14D0  284F            :        JR      Z,DAT1-$
14D2  FE08            :        CP      8               ; LST OPCODE
14D4  C8              :        RET     Z
14D5  FE09            :        CP      9               ; NLST OPCODE
14D7  C8              :        RET     Z
14D8  FE0A            :        CP      10              ; EXEC OPCODE
14DA  C8              :        RET     Z
14DB  FE07            :        CP      7
14DD  2845            :        JR      Z,SASC1-$       ; ASC OPCODE
14DF  FE05            :        CP      5
14E1  3836            :        JR      C,RES1-$        ; RES OPCODE
14E3  C2 2A17         :        JP      NZ,EPASS        ; JUMP IF END
                      :*  DO DW PSEUDO OP
14E6  0E02            :ACO1    LD      C,2             ; 2 BYTE INSTRUCTION
14E8  AF              :        XOR     A               ; GET A ZERO
14E9  C3 1318         :        JP      OCN1            ; ADD VALUE TO PROGRAM CNTR
                      :*   DO Z80 4-BYTE OPCODE
14EC  0E04            :PSZ81   LD      C,4             ; 4 BYTE INSTRUCTION
14EE  AF              :        XOR     A               ; A=0
14EF  C3 1318         :        JP      OCN1            ; ADD VALUE OF PROGRAM COUNTER

PROGRAM ERRORS --     0
```

```
                        ;*  DO ORG PSEUDO OP
14F2  CD B118           ;ORG1    CALL    ASCN            ; GET OPERAND
14F5  3A ECFD           ;        LD      A,(OBUF)        ; FETCH ERROR INDICATOR
14F8  FE20              ;        CP      ' '             ; CHECK FOR AN ERROR
14FA  C0                ;        RET     NZ
14FB  22 68F2           ;        LD      (ASPC),HL       ; STORE NEW ORIGIN
14FE  3A 01FE           ;        LD      A,(IBUF)        ; GET FIRST CHARACTER
1501  FE20              ;        CP      ' '             ; CHECK FOR LABEL
1503  C8                ;        RET     Z               ; NO LABEL
1504  180B              ;        JR      EQUS-$          ; CHANGE LABEL VALUE
                        ;*  DO EQU PSEUDO OP
1506  CD B118           ;EQU1    CALL    ASCN            ; GET OPERAND
1509  3A 01FE           ;        LD      A,(IBUF)        ; FETCH 1ST CHARACTER
150C  FE20              ;        CP      ' '             ; CHECK FOR LABEL
150E  CA 031A           ;        JP      Z,ERRM          ; MISSING LAEL
1511  EB                ;EQUS    EX      DE,HL
1512  2A 66F2           ;        LD      HL,(TABA)       ; SYMBOL TABLE ADDRESS
1515  72                ;        LD      (HL),D          ; STORE LABEL VALUE
1516  23                ;        INC     HL
1517  73                ;        LD      (HL),E
1518  C9                ;        RET
                        ;*  DO DS PSEUDO OP
1519  CD B118           ;RES1    CALL    ASCN            ; GET OPERAND
151C  44                ;        LD      B,H
151D  4D                ;        LD      C,L
151E  C3 E515           ;        JP      RES21           ; ADD VALUE TO PROGRAM COUNTER
                        ;*  DO DB PSEUDO OP
1521  C3 EC15           ;DAT1    JP      DAT2A
                        ;*  DO ASC PSEUDO OP
1524  01 0000           ;SASC1   LD      BC,0            ; SET ZERO VALUE
1527  2A 73F2           ;        LD      HL,(PNTR)       ; GET POINTER TO LINE CHAR
152A  7E                ;        LD      A,(HL)          ; GET CHAR
152B  FE27              ;        CP      27H             ; CHECK FOR SINGLE QUOTE
152D  C2 E515           ;        JP      NZ,RES21
1530  23                ;SASL1   INC     HL              ; GET CHARS
1531  7E                ;        LD      A,(HL)          ; GET CHAR
1532  FE0D              ;        CP      0DH             ; DONE IF <CR>
1534  CA E515           ;        JP      Z,RES21
1537  FE27              ;        CP      27H             ; DONE IF SINGLE QUOTE
1539  CA E515           ;        JP      Z,RES21
153C  03                ;        INC     BC              ; INCR COUNT
153D  18F1              ;        JR      SASL1-$
                        ;*
                        ;* PERFORM PASS 2 OF THE ASSEMBLER
                        ;*
153F  21 EEFD           ;PASS2   LD      HL,OBUF+2       ; SET OUTPUT BUFFER ADDRESS
1542  3A 69F2           ;        LD      A,(ASPC+1)      ; FETCH PC(HIGH)
1545  CD 8502           ;        CALL    BINH            ; CONVERT FOR OUTPUT
1548  23                ;        INC     HL

PROGRAM ERRORS --    0
```

```
1549  3A 68F2           :        LD      A,(ASPC)        ; FETCH PC(LOW)
154C  CD 8502           :        CALL    BINH            ; CONVERT FOR OUTPUT
154F  23                :        INC     HL
1550  22 7CF2           :        LD      (OIND),HL       ; SAVE OUTPUT ADDRESS
1553  CD 5B01           :        CALL    ZBUF            ; CLEAR BUFFER
1556  21 01FE           :        LD      HL,IBUF         ; INITIALIZE LINE POINTER
1559  22 73F2           :        LD      (PNTR),HL       ; SAVE POINTER
155C  7E                :        LD      A,(HL)          ; FETCH FIRST CHARACTER
155D  FE20              :        CP      ' '             ; CHECK FOR LABEL
155F  CA 9B14           :        JP      Z,OPC           ; GET OPCODE
1562  FE2A              :        CP      '*'             ; CHECK FOR COMMENT
1564  C8                :        RET     Z               ; RETURN IF COMMENT
1565  FE3B              :        CP      ';'             ; CHECK FOR ';' COMMENT
1567  C8                :        RET     Z               ; RETURN IF SO
1568  CD 3D18           :        CALL    SLAB            ; SCAN OFF LABEL
156B  DA 231A           :        JP      C,ERRL          ; ERROR IN LABEL
156E  CD B014           :        CALL    LCHK            ; CHECK FOR A BLANK OR COLON
1571  C2 231A           :        JP      NZ,ERRL         ; ERROR IF NOT A BLANK
1574  C3 9B14           :        JP      OPC
                        ;*  PROCESS PSEUDO OPS FOR PASS 2
1577  1A                ;PSU2    LD      A,(DE)
1578  FE14              :        CP      20
157A  304A              :        JR      NC,PSZ82-$      ; DO Z80 4-BYTE OPCODES
157C  B7                :        OR      A               ; SET FLAGS
157D  CA 2A16           :        JP      Z,ORG2          ; ORG OPCODE
1580  FE01              :        CP      1               ; EQUATE PSEUDO OP
1582  CA 1816           :        JP      Z,EQU2
1585  FE02              :        CP      2               ; DB OPCODE
1587  2860              :        JR      Z,DAT2-$
1589  FE08              :        CP      8               ; LST OPCODE
158B  2826              :        JR      Z,LST2-$
158D  FE09              :        CP      9               ; NLST OPCODE
158F  282E              :        JR      Z,NLST2-$
1591  FE0A              :        CP      10              ; EXEC OPCODE
1593  2811              :        JR      Z,EX2-$
1595  FE07              :        CP      7               ; ASCII PSEUDO OP
1597  2859              :        JR      Z,SASC2-$
1599  FE05              :        CP      5
159B  383C              :        JR      C,RES2-$        ; RES OPCODE
159D  C2 2A17           :        JP      NZ,EPASS        ; END OPCODE
                        ;*  DO DW PSEUDO OP
15A0  CD 0B17           :        CALL    IYSG            ; GET VALUE
15A3  C3 E614           :        JP      ACQ1
                        ;*  DO EXEC PSEUDO OP
15A6  CD AE18           ;EX2     CALL    ASBL            ; GET VALUE
15A9  3A ECFD           :        LD      A,(OBUF)        ; CHECK FOR ERROR
15AC  FE20              :        CP      ' '             ; <SP> MEANS NO ERROR
15AE  C0                :        RET     NZ              ; DON'T SET IF ERROR
15AF  22 0BF0           :        LD      (EXADR),HL      ; NEW EXEC ADR

PROGRAM ERRORS --      0
```

```
1582  C9                :         RET
                        :*  DO LST PSEUDO OP
1583  3A F9F1           :LST2     LD        A,(SPCHAR)      ; SAVE CURRENT LIST OPTION
1586  32 64F2           :         LD        (LSAV),A
1589  3E46              :         LD        A,'F'           ; LIST FORMATTED
158B  32 F9F1           :         LD        (SPCHAR),A
158E  C9                :         RET
                        :*  DO NLST PSEUDO OP
158F  3A 64F2           :NLST2    LD        A,(LSAV)        ; RESTORE CURRENT LIST OPTION
1592  32 F9F1           :         LD        (SPCHAR),A
1595  C9                :         RET
                        :*  DO Z80 4-BYTE OPCODES
1596  F5                :PSZ82    PUSH      AF              ; STORE OPCODE
1597  3EED              :         LD        A,0EDH          ; STORE FIRST BYTE
1599  CD 1717           :         CALL      ASTO
159C  F1                :         POP       AF
159D  CD 1717           :         CALL      ASTO            ; STORE SECOND BYTE
15D0  CD 0B17           :         CALL      TYS6
15D3  0E04              :         LD        C,4             ; 4 BYTE INSTRUCTION
15D5  AF                :         XOR       A               ; SET FOR RETURN
15D6  C3 1318           :         JP        OCN1
                        :*  DO DS PSEUDO OP
15D9  CD AE18           :RES2     CALL      ASBL            ; GET OPERAND
15DC  44                :         LD        B,H
15DD  4D                :         LD        C,L
15DE  2A 0FF2           :         LD        HL,(BBUF+2)     ; FETCH STORAGE COUNTER
15E1  09                :         ADD       HL,BC           ; ADD VALUE
15E2  22 0FF2           :         LD        (BBUF+2),HL
15E5  AF                :RES21    XOR       A               ; GET A ZERO
15E6  C3 1618           :         JP        OCN2
                        :*  DO DB PSEUDO OP
15E9  CD CB16           :DAT2     CALL      TYS5            ; GET OPERAND
15EC  AF                :DAT2A    XOR       A               ; MAKE A ZERO
15ED  0E01              :         LD        C,1             ; BYTE COUNT
15EF  C3 1318           :         JP        OCN1
                        :*  DO ASC PSEUDO OP
15F2  CD 7903           :ASC2     CALL      SBLK            ; SKIP TO FIRST NON-BLANK CHAR
15F5  01 0000           :         LD        BC,0            ; SET COUNT
15F8  2A 73F2           :         LD        HL,(PNTR)       ; GET PTR TO CHAR
15FB  7E                :         LD        A,(HL)          ; GET 1ST BYTE
15FC  FE27              :         CP        27H             ; ERROR IF NOT SINGLE QUOTE
15FE  C4 F119           :         CALL      NZ,ERRS
1601  23                :ASL2     INC       HL              ; GET AND STORE CHARS
1602  7E                :         LD        A,(HL)          ; CHECK FOR EOL
1603  FE0D              :         CP        0DH             ; DONE IF <CR>
1605  28DE              :         JR        Z,RES21-$
1607  FE27              :         CP        27H             ; DONE IF SINGLE QUOTE
1609  28DA              :         JR        Z,RES21-$
160B  EB                :         EX        DE,HL           ; SAVE H&L IN D&E

PROGRAM ERRORS --     0
```

```
160C  2A 0FF2            :         LD      HL,(BBUF+2)     ; GET ADR PTR
160F  77                  :         LD      (HL),A          ; STORE CHAR
1610  23                  :         INC     HL              ; INCR ADR
1611  22 0FF2            :         LD      (BBUF+2),HL     ; SAVE ADR PTR
1614  EB                  :         EX      DE,HL           ; RESTORE H&L
1615  03                  :         INC     BC              ; INCR COUNT
1616  18E9                :         JR      SASL2-$
                          :*  HANDLE EQUATES ON 2ND PASS
1618  CD AE18             :EQU2     CALL    ASBL            ; GET OPERAND INTO HL AND FALL INTO NEXT ROUTINE
                          :*
                          :* STORE CONTENTS OF HL AS HEX ASCII AT OBUF+2.
                          :*  ON RETURN, DE HOLDS VALUE WHICH WAS IN HL.
                          :*
161B  EB                  :BINAD    EX      DE,HL           ; PUT VALUE INTO DE
161C  21 EEFD             :         LD      HL,OBUF+2       ; POINTER TO ADDR IN OBUF
161F  7A                  :         LD      A,D             ; STORE HI BYTE...
1620  CD 8502             :         CALL    BINH
1623  23                  :         INC     HL
1624  7B                  :         LD      A,E             ; STORE LO BYTE...
1625  CD 8502             :         CALL    BINH
1628  23                  :         INC     HL
1629  C9                  :         RET
                          :*  DO ORG PSEUDO OP
162A  CD AE18             :ORG2     CALL    ASBL            ; GET NEW ORIGIN
162D  3A ECFD             :         LD      A,(OBUF)        ; GET ERROR INDICATOR
1630  FE20                :         CP      ' '             ; CHECK FOR AN ERROR
1632  C0                  :         RET     NZ              ; DON'T MODIFY PC IF ERROR
1633  CD 1B16             :         CALL    BINAD           ; STORE NEW ADDR IN OBUF
1636  2A 68F2             :         LD      HL,(ASPC)       ; FETCH PC
1639  EB                  :         EX      DE,HL
163A  22 68F2             :         LD      (ASPC),HL       ; STORE NEW PC
163D  22 6AF2             :         LD      (ASMST),HL      ; REDEFINE START OF ASSEMBLY
1640  7D                  :         LD      A,L
1641  93                  :         SUB     E               ; FORM DIFFERENCE OF ORIGINS
1642  5F                  :         LD      E,A
1643  7C                  :         LD      A,H
1644  9A                  :         SBC     A,D
1645  57                  :         LD      D,A
1646  2A 0FF2             :         LD      HL,(BBUF+2)     ; FETCH STORAGE POINTER
1649  19                  :         ADD     HL,DE           ; MODIFY
164A  22 0FF2             :         LD      (BBUF+2),HL     ; SAVE
164D  C9                  :         RET
                          :*  PROCESS 1 BYTE INSTRUCTIONS WITHOUT OPERANDS
164E  C3 1717             :TYP1     JP      ASTO            ; STORE VALUE IN MEMORY
                          :*  PROCESS STAX AND LDAX INSTRUCTIONS
1651  CD AE18             :TYP2     CALL    ASBL            ; FETCH OPERAND
1654  C4 E819             :         CALL    NZ,ERRR         ; ILLEGAL REGISTER
1657  7D                  :         LD      A,L             ; GET LOW ORDER OPERAND
1658  B7                  :         OR      A               ; SET FLAGS
```

PROGRAM ERRORS -- 0

```
1659  2818            :        JR      Z,TY31-$        ; OPERAND = 0
165B  FE02            :        CP      2               ; OPERAND = 2
165D  C4 E819         :        CALL    NZ,ERRR         ; ILLEGAL REGISTER
1660  1811            :        JR      TY31-$
                      :* PROCESS PUSH, POP, INX, DCX, DAD INSTRUCTIONS
1662  CD AE18         :TYP3    CALL    ASBL            ; FETCH OPERAND
1665  C4 E819         :        CALL    NZ,ERRR         ; ILLEGAL REGISTER
1668  7D              :        LD      A,L             ; GET LOW ORDER OPERAND
1669  0F              :        RRCA                    ; CHECK LOW ORDER BIT
166A  DC E819         :        CALL    C,ERRR          ; ILLEGAL REGISTER
166D  07              :        RLCA                    ; RESTORE
166E  FE08            :        CP      8
1670  D4 E819         :        CALL    NC,ERRR         ; ILLEGAL REGISTER
1673  07              :TY31    RLCA                    ; MULTIPLY BY 8
1674  07              :        RLCA
1675  07              :        RLCA
1676  47              :TY32    LD      B,A
1677  1A              :        LD      A,(DE)          ; FETCH OPCODE BASE
1678  80              :        ADD     A,B             ; FORM OPCODE
1679  FE76            :        CP      118             ; CHECK FOR LD (HL),(HL)
167B  CC E819         :        CALL    Z,ERRR          ; ILLEGAL REGISTER
167E  18CE            :        JR      TYP1-$
                      :* PROCESS ACCUMULATOR, INC, DEC, MOV, RST INSTRUCTIONS
1680  CD AE18         :TYP4    CALL    ASBL            ; FETCH OPERAND
1683  C4 E819         :        CALL    NZ,ERRR         ; ILLEGAL REGISTER
1686  7D              :        LD      A,L             ; GET LOW ORDER OPERAND
1687  FE08            :        CP      8
1689  D4 E819         :        CALL    NC,ERRR         ; ILLEGAL REGISTER
168C  1A              :        LD      A,(DE)          ; FETCH OPCODE BASE
168D  FE40            :        CP      64              ; CHECK FOR LD  INSTRUCTION
168F  280A            :        JR      Z,TY41-$
1691  FEC7            :        CP      199
1693  7D              :        LD      A,L
1694  28DD            :        JR      Z,TY31-$        ; RST INSTRUCTION
1696  FA 7616         :        JP      M,TY32          ; ACCUMULATOR INSTRUCTION
1699  18D8            :        JR      TY31-$          ; INC, DEC
                      :* PROCESS MOV INSTRUCTION
169B  29              :TY41    ADD     HL,HL           ; MULTIPLY OPERAND BY 8
169C  29              :        ADD     HL,HL
169D  29              :        ADD     HL,HL
169E  85              :        ADD     A,L             ; FORM OPCODE
169F  12              :        LD      (DE),A          ; SAVE OPCODE
16A0  CD EA16         :        CALL    MPNT
16A3  CD B118         :        CALL    ASCN
16A6  C4 E819         :        CALL    NZ,ERRR         ; INCREMENT POINTER
16A9  7D              :        LD      A,L             ; FETCH LOW ORDER OPERAND
16AA  FE08            :        CP      8
16AC  D4 E819         :        CALL    NC,ERRR         ; ILLEGAL REGISTER
16AF  18C5            :        JR      TY32-$

PROGRAM ERRORS --      0
```

```
                        :*
                        :* PROCESS IMMEDIATE INSTRUCTIONS.
                        :*  IMMEDIATE BYTE CAN BE BETWEEN -128 AND +127.
                        :*  MVI INSTRUCTION IS A SPECIAL CASE AND CONTAINS
                        :*  2 ARGUMENTS IN OPERAND.
                        :*
  16B1  FE39            :TYP5   CP      57              ; CHECK FOR MVI AND Z80 RELATIVE BRANCHES
  16B3  380E            :       JR      C,TYP5A-$       ; HANDLE MVI
  16B5  FEC6            :       CP      198             ; CHECK FOR Z80 MNEUMONICS
  16B7  300A            :       JR      NC,TYP5A-$      ; HANDLE NORMAL 8080 MNEUMONICS
  16B9  F5              :       PUSH    AF              ; SAVE OP CODE
  16BA  3EED            :       LD      A,0EDH          ; SET UP SPECIAL Z80 OPCODE BYTE
  16BC  CD 1717         :       CALL    ASTO            ; PROCESS AS OPCODE
  16BF  F1              :       POP     AF              ; GET SECOND PART OF OPCODE
  16C0  C3 1717         :       JP      ASTO            ; PROCESS
  16C3  FE06            :TYP5A  CP      6               ; CHECK FOR MVI INSTRUCTION
  16C5  CC D816         :       CALL    Z,TY56
  16C8  CD 1717         :       CALL    ASTO            ; STORE OBJECT BYTE
  16CB  CD AE18         :TYS5   CALL    ASBL            ; GET IMMEDIATE ARGUMENT
  16CE  3C              :       INC     A
  16CF  FE02            :       CP      2               ; CHECK OPERAND FOR RANGE
  16D1  D4 FF19         :       CALL    NC,ERRV         ; OPERAND OUT OF RANGE
  16D4  7D              :       LD      A,L
  16D5  C3 4E16         :       JP      TYP1
                        :*  FETCH 1ST ARG FOR LD  AND LXI INSTRUCTIONS
  16D8  CD AE18         :TY56   CALL    ASBL            ; FETCH ARG
  16DB  C4 E819         :       CALL    NZ,ERRR         ; ILLEGAL REGISTER
  16DE  7D              :       LD      A,L             ; GET LOW ORDER ARGUMENT
  16DF  FE08            :       CP      8
  16E1  D4 E819         :       CALL    NC,ERRR         ; ILLEGAL REGISTER
  16E4  29              :       ADD     HL,HL
  16E5  29              :       ADD     HL,HL
  16E6  29              :       ADD     HL,HL
  16E7  1A              :       LD      A,(DE)          ; FETCH OPCODE BASE
  16E8  85              :       ADD     A,L             ; FOR OPCODE
  16E9  5F              :       LD      E,A             ; SAVE OBJECT BYTE
  16EA  2A 73F2         :MPNT   LD      HL,(PNTR)       ; FETCH POINTER
  16ED  7E              :       LD      A,(HL)          ; FETCH CHARACTER
  16EE  FE2C            :       CP      ','             ; CHECK FOR COMMA
  16F0  23              :       INC     HL              ; INCREMENT POINTER
  16F1  22 73F2         :       LD      (PNTR),HL
  16F4  C2 F119         :       JP      NZ,ERRS         ; SYNTAX ERROR IF NO COMMA
  16F7  7B              :       LD      A,E
  16F8  C9              :       RET
                        :*
                        :* PROCESS 3 BYTE INSTRUCTIONS.
                        :*  LXI IS  SPECIAL CASE.
                        :*
  16F9  FE01            :TYP6   CP      1               ; CHECK FOR LXI INSTRUCTION
```

PROGRAM ERRORS -- 0

```
16FB  200B            :        JR      NZ,TY6-$        ; JUMP IF NOT LXI
16FD  CD D816         :        CALL    TY56            ; GET REGISTER
1700  E60B            :        AND     0BH             ; CHECK FOR ILLEGAL REGISTER
1702  C4 E819         :        CALL    NZ,ERRR         ; REGISTER ERROR
1705  7B              :        LD      A,E             ; GET OPCODE
1706  E6F7            :        AND     0F7H            ; CLEAR BIT IN ERROR
1708  CD 1717         :TY6     CALL    ASTO            ; STORE OBJECT BYTE
170B  CD AE18         :TY56    CALL    ASBL            ; FETCH OPERAND
170E  7D              :        LD      A,L
170F  54              :        LD      D,H
1710  CD 1717         :        CALL    ASTO            ; STORE 2ND BYTE
1713  7A              :        LD      A,D
1714  C3 4E16         :        JP      TYP1
                      :*
                      :* THIS ROUTINE IS USED TO STORE OBJECT CODE PRODUCED
                      :*  BY THE ASSEMBLER DURING PASS 2 INTO MEMORY.
                      :*
1717  2A 0FF2         :ASTO    LD      HL,(BBUF+2)     ; FETCH STORAGE ADDRESS
171A  77              :        LD      (HL),A          ; STORE OBJECT BYTE
171B  23              :        INC     HL              ; PT TO NEXT MEMORY LOC
171C  22 0FF2         :        LD      (BBUF+2),HL
171F  2A 7CF2         :        LD      HL,(OIND)       ; FETCH OUTPUT ADDRESS
1722  23              :        INC     HL
1723  CD 8502         :        CALL    BINH            ; CONVERT OBJECT BYTE
1726  22 7CF2         :        LD      (OIND),HL
1729  C9              :        RET
                      :*
                      :* GET HERE WHEN END PSEUDO OP IS FOUND OR WHEN
                      :*  END-OF-FILE OCCURS IN SOURCE FILE.  CONTROL IS SET
                      :*  FOR EITHER PASS 2 OR ASSEMBLY TERMINATOR IF FINISHED.
                      :*
172A  3A 71F2         :EPASS   LD      A,(PASS)        ; FETCH PASS INDICATOR
172D  B7              :        OR      A               ; SET FLAGS
172E  2012            :        JR      NZ,EPASS1-$     ; JUMP IF FINISHED
1730  2A 68F2         :        LD      HL,(ASPC)       ; GET END PC
1733  2B              :        DEC     HL              ; ADJUST
1734  22 6CF2         :        LD      (ASMEN),HL
1737  21 661C         :        LD      HL,MSG1         ; 'PASS 2' MSG
173A  CD 8A03         :        CALL    SCRN
173D  3E01            :        LD      A,1             ; PASS INDICATOR FOR 2ND PASS
173F  C3 BC13         :        JP      ASM2            ; DO 2ND PASS
1742  CD 3FD0         :EPASS1  CALL    CRLF            ; PRINT LIMITS OF ASSEMBLY
1745  2A 6AF2         :        LD      HL,(ASMST)
1748  CD 6FD0         :        CALL    PHLB            ; PRINT START ADR
174B  EB              :        EX      DE,HL
174C  2A 6CF2         :        LD      HL,(ASMEN)
174F  CD 6FD0         :        CALL    PHLB            ; PRINT END ADR
1752  EB              :        EX      DE,HL
1753  CD 9FD0         :        CALL    RANGE           ; COMPUTE RANGE

PROGRAM ERRORS --     0
```

```
1756  69              :        LD      L,C             : PRINT RANGE
1757  60              :        LD      H,B
1758  CD 6FD0         :        CALL    PHLB
175B  CD 9F02         :        CALL    CKFN1           : FILE NAME OF BINARY FILE?
175E  2803            :        JR      Z,EPASS2-$      : NO ENTRY
1760  CD B812         :        CALL    BFDENT          : MAKE ENTRY IN BF DIR OF (ASMST), (ASMEN)
1763  3A 6EF2         :EPASS2  LD      A,(SYSERR)      : CHECK FOR ERROR
1766  B7              :        OR      A               : ERROR IF A=1
1767  C2 BA00         :        JP      NZ,EOR          : RETURN TO MAIN SYSTEM IF ERROR
176A  2A 6FF2         :        LD      HL,(ASMRET)     : GET RETURN ADDRESS
176D  E9              :        JP      (HL)
                      :*
                      :* THIS ROUTINE IS USED TO CHECK THE CONDITION
                      :*  CODE MNEMONICS FOR CONDITIONAL JUMPS, CALLS,
                      :*  AND RETURNS.
                      :*
176E  21 FEF1         :COND    LD      HL,ABUF+1
1771  22 E5F1         :        LD      (ADDS),HL
1774  0602            :        LD      B,2             : 2 CHARACTERS
1776  C3 7917         :        JP      COPC
                      :*
                      :* THIS ROUTINE IS USED TO CHECK A GIVEN OPCODE
                      :*  AGAINST THE LEGAL OPCODES IN THE OPCODE TABLE.
                      :*
1779  2A E5F1         :COPC    LD      HL,(ADDS)
177C  1A              :        LD      A,(DE)          : FETCH CHARACTER
177D  B7              :        OR      A               : SET FLAGS
177E  2809            :        JR      Z,COP1-$        : JUMP IF TERMINATION CHARACTER
1780  48              :        LD      C,B
1781  CD 4201         :        CALL    SEAR
1784  1A              :        LD      A,(DE)
1785  C8              :        RET     Z               : RETURN IF MATCH
1786  13              :        INC     DE              : NEXT STRING
1787  18F0            :        JR      COPC-$          : CONTINUE SEARCH
1789  3C              :COP1    INC     A               : CLEAR ZERO FLAG
178A  13              :        INC     DE              : INCREMENT ADDRESS
178B  C9              :        RET
                      :*
                      :* THIS ROUTINE CHECKS THE LEGAL OPCODES IN BOTH PASS 1
                      :*  AND PASS 2.  IN PASS 1, THE PROGRAM COUNTER IS INCREMENTED
                      :*  BY THE CORRECT NUMBER OF BYTES.  AN ADDRESS IS
                      :*  ALSO SET SO THAT AN INDEXED JUMP CAN BE MADE TO
                      :*  PROCESS THE OPCODE FOR PASS 2.
                      :*
178C  21 FDF1         :OPCD    LD      HL,ABUF         : GET ADDRESS
178F  22 E5F1         :        LD      (ADDS),HL
1792  11 F71D         :        LD      DE,OTAB         : OPCODE TABLE ADDRESS
1795  0604            :        LD      B,4             : CHARACTER COUNT
1797  CD 7917         :        CALL    COPC            : CHECK OPCODES

PROGRAM ERRORS --  0
```

```
179A  CA 2B18           :        JP      Z,PSEU          ; JUMP IF A PSEUDO OP
179D  05                :        DEC     B               ; 3 CHARACTER OPCODES
179E  CD 7917           :        CALL    COPC
17A1  2804              :        JR      Z,OP1-$
17A3  04                :        INC     B               ; 4 CHARACTER OPCODES
17A4  CD 7917           :        CALL    COPC
17A7  21 4E16           :OP1     LD      HL,TYP1         ; TYPE 1 INSTRUCTION
17AA  0E01              :OP2     LD      C,1             ; 1 BYTE INSTRUCTIONS
17AC  2851              :        JR      Z,OCNT-$
17AE  CD 7917           :        CALL    COPC            ; CHECK FOR STAX, LDAX
17B1  21 5116           :        LD      HL,TYP2
17B4  28F4              :        JR      Z,OP2-$
17B6  CD 7917           :        CALL    COPC            ; CHECK FOR PUSH, POP, INX, DCX, DAD
17B9  21 6216           :        LD      HL,TYP3
17BC  28EC              :        JR      Z,OP2-$
17BE  05                :        DEC     B               ; 3 CHARACTER OPCODES
17BF  CD 7917           :        CALL    COPC            ; ACC INSTRUCTIONS, INC, DEC, MOV, RST
17C2  21 8016           :        LD      HL,TYP4
17C5  28E3              :        JR      Z,OP2-$
17C7  CD 7917           :        CALL    COPC            ; IMMEDIATE INSTRUCTIONS
17CA  21 B116           :        LD      HL,TYP5
17CD  0E02              :        LD      C,2
17CF  282E              :        JR      Z,OCNT-$
17D1  04                :        INC     B               ; 4 CHAR OPCODES
17D2  CD 7917           :        CALL    COPC            ; JMP,CALL,LXI,LDA,STA,LHLD,SHLD
17D5  2823              :        JR      Z,OP4-$
17D7  CD 6E17           :        CALL    COND            ; CONDITIONAL INSTRUCTIONS
17DA  C2 2418           :        JP      NZ,OERR         ; ILLEGAL OPCODE
17DD  C6C0              :        ADD     A,192           ; ADD BASE VALUE TO RETURN
17DF  57                :        LD      D,A
17E0  0603              :        LD      B,3             ; 3 CHAR OPCODES
17E2  3A FDF1           :        LD      A,(ABUF)        ; FETCH FIRST CHAR
17E5  4F                :        LD      C,A             ; SAVE CHAR
17E6  FE52              :        CP      'R'             ; CONDITIONAL RETURN
17E8  7A                :        LD      A,D
17E9  288C              :        JR      Z,OP1-$
17EB  79                :        LD      A,C
17EC  14                :        INC     D               ; FORM CONDITIONAL JUMP
17ED  14                :        INC     D
17EE  FE4A              :        CP      'J'             ; CONDITIONAL JUMP
17F0  2807              :        JR      Z,OPAD-$
17F2  FE43              :        CP      'C'             ; CONDITIONAL CALL
17F4  C2 2418           :        JP      NZ,OERR         ; ILLEGAL OPCODE
17F7  14                :        INC     D               ; FORM CONDITIONAL CALL
17F8  14                :        INC     D
17F9  7A                :OPAD    LD      A,D             ; GET OPCODE
17FA  21 F916           :OP4     LD      HL,TYP6
17FD  0E03              :OP5     LD      C,3             ; 3 BYTE INSTRUCTION
17FF  32 7BF2           :OCNT    LD      (TEMP),A        ; SAVE OPCODE

PROGRAM ERRORS --      0
```

```
                        ;*
                        ;* CHECK FOR OPCODE ONLY CONTAINING THE CORRECT NUMBER OF
                        ;*  CHARACTERS.  THUS ADDQ, SAY, WOULD GIVE AN ERROR.
                        ;*
1802  3EFD              :       LD      A,ABUF          ; LOAD BUFFER ADDRESS
1804  80                :       ADD     A,B             ; ADD LENGTH OF OPCODE
1805  5F                :       LD      E,A
1806  3EF1              :       LD      A,ABUF/256
1808  CE00              :       ADC     A,0             ; GET HIGH ORDER ADDRESS
180A  57                :       LD      D,A
180B  1A                :       LD      A,(DE)          ; FETCH CHARACTER AFTER OPCODE
180C  B7                :       OR      A               ; IT SHOULD BE ZERO
180D  C2 2418           :       JP      NZ,OERR         ; OPCODE ERROR
1810  3A 71F2           :OCN0   LD      A,(PASI)        ; FETCH PASS INDICATOR
1813  0600              :OCN1   LD      B,0
1815  EB                :       EX      DE,HL
1816  2A 68F2           :OCN2   LD      HL,(ASPC)       ; FETCH PROGRAM COUNTER
1819  09                :       ADD     HL,BC           ; ADD IN BYTE COUNT
181A  22 68F2           :       LD      (ASPC),HL       ; STORE PC
181D  B7                :       OR      A               ; WHICH PASS?
181E  C8                :       RET     Z               ; RETURN IF PASS 1
181F  3A 7BF2           :       LD      A,(TEMP)        ; FETCH OPCODE
1822  EB                :       EX      DE,HL
1823  E9                :       JP      (HL)
1824  21 0F1A           :OERR   LD      HL,ERRO         ; SET ERROR ADDRESS
1827  0E03              :       LD      C,3             ; LEAVE 3 BYTES FOR PATCH
1829  18E5              :       JR      OCN0-$
182B  21 01F2           :PSEU   LD      HL,ABUF+4       ; SET BUFFER ADDRESS
182E  7E                :       LD      A,(HL)          ; FETCH CHARACTER AFTER OPCODE
182F  B7                :       OR      A               ; SHOULD BE A ZERO
1830  C2 2418           :       JP      NZ,OERR
1833  3A 71F2           :       LD      A,(PASI)        ; FETCH PASS INDICATOR
1836  B7                :       OR      A
1837  CA BF14           :       JP      Z,PSU1
183A  C3 7715           :       JP      PSU2
                        ;*
                        ;* THIS ROUTINE IS USED TO PROCESS LABELS.
                        ;*  IT CHECKS TO SEE IF A LABEL IS IN THE SYMBOL TABLE
                        ;*  OR NOT.  ON RETURN, Z=1 INDICATES A MATCH WAS FOUND
                        ;*  AND H,L CONTAIN THE VALUE ASSOCIATED WITH THE LABEL.
                        ;*  THE REGISTER NAMES A,B,C,D,E,H,L,P, AND S ARE
                        ;*  PRE-DEFINED AND NEED NOT BE ENTERED BY THE USER.
                        ;*  ON RETURN, C=1 INDICATES A LABEL ERROR.
                        ;*
183D  FE41              :GLAB   CP      'A'             ; CHECK FOR LEGAL CHAR
183F  D8                :       RET     C
1840  FE5B              :       CP      'Z'+1           ; CHECK FOR ILLEGAL CHAR
1842  3F                :       CCF
1843  D8                :       RET     C

PROGRAM ERRORS --       0
```

```
1844  CD 8B18           :         CALL    ALPS            ; PLACE SYMBOL IN BUFFER
1847  21 FDF1           :         LD      HL,ABUF         ; SET BUFFER ADDRESS
184A  22 E5F1           :         LD      (ADDS),HL       ; SAVE ADDRESS
184D  100E              :         DJNZ    SLA1-$
                        :*  CHECK IF PREDEFINED REGISTER NAME
184F  04                :         INC     B               ; SET B=1
1850  11 EA1F           :         LD      DE,RTAB         ; REGISTER TABLE ADDRESS
1853  CD 7917           :         CALL    COPC            ; CHECK NAME OF REGISTER
1856  2005              :         JR      NZ,SLA1-$       ; NOT A PREDEFINED REGISTER
1858  6F                :         LD      L,A             ; SET VALUE (HIGH)
1859  2600              :         LD      H,0
185B  1829              :         JR      SLA2-$
185D  ED4B 75F2         :SLA1     LD      BC,(NOLA)       ; CHECK FOR NO LABELS
1861  78                :         LD      A,B
1862  B1                :         OR      C
1863  2824              :         JR      Z,SLA3-$
                        :*    SYMBOL TABLE SEARCH
1865  21 FDF1           :         LD      HL,ABUF         ; SEARCH FOR NAME IN ABUF
1868  11 00F3           :SCANL    LD      DE,SYMT         ; GENERAL SYMTABLE SCAN; RET ZERO = FOUND
186B  C5                :SRCHST   PUSH    BC              ; SAVE BC, HL
186C  E5                :         PUSH    HL
186D  0E06              :         LD      C,LLAB          ; C=LENGTH OF LABEL
186F  CD 4201           :         CALL    SEAR            ; COMPARE
1872  1A                :         LD      A,(DE)          ; STORE SYMBOL VALUE IN HL
1873  67                :         LD      H,A
1874  13                :         INC     DE
1875  1A                :         LD      A,(DE)
1876  6F                :         LD      L,A
1877  13                :         INC     DE
1878  280A              :         JR      Z,SRCHQT-$
187A  E1                :         POP     HL
187B  C1                :         POP     BC
187C  0B                :         DEC     BC
187D  78                :         LD      A,B
187E  B1                :         OR      C               ; DONE?
187F  20EA              :         JR      NZ,SRCHST-$
1881  04                :         INC     B
1882  1802              :         JR      SLA2-$
1884  C1                :SRCHQT   POP     BC              ; CLEAR STACK
1885  C1                :         POP     BC
1886  37                :SLA2     SCF                     ; SET CARRY
1887  3F                :         CCF                     ; CLEAR CARRY
1888  C9                :         RET                     ; RETURN
1889  3C                :SLA3     INC     A               ; CLEAR ZERO FLAG
188A  C9                :         RET
                        :*
                        :* THIS ROUTINE SCANS THE INPUT LINE AND PLACES THE
                        :*  OPCODES AND LABELS IN THE BUFFER.  THE SCAN TERMINATES
                        :*  WHEN A CHARACTER OTHER THAN 0-9 OR A-Z IS FOUND.
```

PROGRAM ERRORS -- 0

```
                        ;*
188B  0600              :ALPS   LD      B,0             ; SET COUNT
188D  CD 05D1           :       CALL    CAPS            ; CAPITALIZE
1890  12                :ALP1   LD      (DE),A          ; STORE CHARACTER IN BUFFER
1891  04                :       INC     B               ; INCREMENT COUNT
1892  78                :       LD      A,B             ; FETCH COUNT
1893  FE0F              :       CP      15              ; MAXIMUM BUFFER SIZE
1895  D0                :       RET     NC              ; RETURN IF BUFFER FILLED
1896  13                :       INC     DE              ; INCREMENT BUFFER
1897  23                :       INC     HL              ; INCREMENT INPUT POINTER
1898  22 73F2           :       LD      (PNTR),HL       ; SAVE LINE POINTER
189B  7E                :       LD      A,(HL)          ; FETCH CHARACTER
189C  CD 05D1           :       CALL    CAPS            ; CAPITALIZE
189F  FE30              :       CP      '0'             ; CHECK FOR LEGAL CHARACTERS
18A1  D8                :       RET     C
18A2  FE3A              :       CP      '9'+1
18A4  38EA              :       JR      C,ALP1-$
18A6  FE41              :       CP      'A'
18A8  D8                :       RET     C
18A9  FE5B              :       CP      'Z'+1
18AB  38E3              :       JR      C,ALP1-$
18AD  C9                :       RET
                        ;*
                        ;* THIS ROUTINE IS USED TO SCAN THROUGH THE INPUT LINE
                        ;*  TO FETCH THE VALUE OF THE OPERAND FIELD.  ON RETURN,
                        ;*  THE VALUE OF THE OPERAND IS CONTAINED IN REG'S H,L.
                        ;*
18AE  CD 7903           :ASBL   CALL    SBLK            ; GET FIRST ARGUMENT
18B1  21 0000           :ASCN   LD      HL,0            ; GET A ZERO
18B4  22 78F2           :       LD      (OPRD),HL       ; INITIALIZE OPERAND
18B7  24                :       INC     H
18B8  22 79F2           :       LD      (OPRI-1),HL     ; INITIALIZE OPERAND INDICATOR
18BB  2A 73F2           :NXT1   LD      HL,(PNTR)       ; FETCH SCAN POINTER
18BE  2B                :       DEC     HL
18BF  CD 5B01           :       CALL    ZBUF            ; CLEAR BUFFER
18C2  32 77F2           :       LD      (OPER),A        ; ZERO OPERATOR INDICATOR
18C5  23                :NXT2   INC     HL              ; INCREMENT POINTER
18C6  7E                :       LD      A,(HL)          ; FETCH NEXT CHARACTER
18C7  FE21              :       CP      ' '+1
18C9  DA BE19           :       JP      C,SEND          ; JUMP IF CR OR BLANK
18CC  FE2C              :       CP      ','             ; FIELD SEPARATOR
18CE  CA BE19           :       JP      Z,SEND
                        ;*
                        ;*  CHECK FOR OPERATORS
                        ;*    VALID OPERATORS ARE --
                        ;*       + FOR ADDITION
                        ;*       - FOR SUBTRACTION
                        ;*       ! FOR LOGICAL OR
                        ;*       & FOR LOGICAL AND

PROGRAM ERRORS --       0
```

```
                        :*      > FOR EXTRACT LOW (ADD ARGS & MASK OUT HIGH BYTE)
                        :*      < FOR EXTRACT HIGH (ADD ARGS, MASK OUT LOW BYTE, AND
                        :*        MOVE HIGH BYTE TO LOW BYTE)
                        :*      % FOR LOGICAL EXCLUSIVE OR
                        :*      * FOR INTEGER MULTIPLY
                        :*      / FOR INTEGER DIVIDE
                        :*
18D1  FE2B              :       CP      '+'             ; CHECK FOR PLUS
18D3  2823              :       JR      Z,ASC1-$
18D5  FE2D              :       CP      '-'             ; CHECK FOR MINUS
18D7  281C              :       JR      Z,ASC0-$
18D9  FE3E              :       CP      '>'             ; EXTRACT LOW?
18DB  2818              :       JR      Z,ASC0-$
18DD  FE3C              :       CP      '<'             ; EXTRACT HIGH?
18DF  2814              :       JR      Z,ASC0-$
18E1  FE21              :       CP      '!'             ; LOGICAL OR
18E3  2810              :       JR      Z,ASC0-$
18E5  FE25              :       CP      '%'             ; LOGICAL XOR
18E7  280C              :       JR      Z,ASC0-$
18E9  FE26              :       CP      '&'             ; LOGICAL AND
18EB  2808              :       JR      Z,ASC0-$
18ED  FE2A              :       CP      '*'             ; MULTIPLY
18EF  2804              :       JR      Z,ASC0-$
18F1  FE2F              :       CP      '/'             ; DIVIDE
18F3  2012              :       JR      NZ,ASC2-$
18F5  32 77F2           :ASC0   LD      (OPER),A
18F8  3A 7AF2           :ASC1   LD      A,(OPRI)        ; FETCH OPERAND INDICATOR
18FB  FE02              :       CP      2               ; CHECK FOR TWO OPERATORS
18FD  CA F119           :       JP      Z,ERRS          ; SYNTAX ERROR
1900  3E02              :       LD      A,2
1902  32 7AF2           :       LD      (OPRI),A        ; SET INDICATOR
1905  18BE              :       JR      NXT2-$
                        :*  CHECK FOR OPERANDS
1907  4F                :ASC2   LD      C,A             ; SAVE CHAR
1908  3A 7AF2           :       LD      A,(OPRI)        ; GET INDICATOR
190B  B7                :       OR      A               ; CHECK FOR 2 OPERANDS
190C  CA F119           :       JP      Z,ERRS          ; SYNTAX ERROR
190F  79                :       LD      A,C
1910  FE24              :       CP      '$'             ; LC EXPRESSION
1912  2009              :       JR      NZ,ASC3-$
1914  23                :       INC     HL              ; INCREMENT POINTER
1915  22 73F2           :       LD      (PNTR),HL       ; SAVE POINTER
1918  2A 68F2           :       LD      HL,(ASPC)       ; FETCH LOCATION COUNTER
191B  1837              :       JR      AVAL-$
                        :*  CHECK FOR ASCII CHARACTERS
191D  FE27              :ASC3   CP      27H             ; CHECK FOR SINGLE QUOTE
191F  2024              :       JR      NZ,ASC5-$       ; JUMP IF NOT QUOTE
1921  11 0000           :       LD      DE,0            ; GET A ZERO
1924  0E03              :       LD      C,3             ; CHARACTER COUNT

PROGRAM ERRORS --       0
```

```
1926  23              :ASC4   INC     HL              ; BUMP POINTER
1927  22 73F2         :       LD      (PNTR),HL       ; SAVE
192A  7E              :       LD      A,(HL)          ; FETCH NEXT CHAR
192B  FE0D            :       CP      0DH             ; IS IT A CR?
192D  CA 0B1A         :       JP      Z,ERRA          ; ARGUMENT ERROR
1930  FE27            :       CP      27H             ; IS IT A QUOTE?
1932  2009            :       JR      NZ,SSTR-$
1934  23              :       INC     HL              ; INCREMENT POINTER
1935  22 73F2         :       LD      (PNTR),HL       ; SAVE
1938  7E              :       LD      A,(HL)          ; FETCH NEXT CHAR
1939  FE27            :       CP      27H             ; CHECK FOR 2 QUOTES IN A ROW
193B  2018            :       JR      NZ,AVAL1-$      ; TERMINAL QUOTE
193D  0D              :SSTR   DEC     C               ; CHECK COUNT
193E  CA 0B1A         :       JP      Z,ERRA          ; TOO MANY CHARS
1941  53              :       LD      D,E
1942  5F              :       LD      E,A             ; SET CHAR IN BUFFER
1943  18E1            :       JR      ASC4-$
1945  FE30            :ASC5   CP      '0'             ; CHECK FOR NUMERIC
1947  DA 0B1A         :       JP      C,ERRA          ; ILLEGAL CHAR
194A  FE3A            :       CP      '9'+1
194C  3067            :       JR      NC,ALAB-$
194E  CD CE19         :       CALL    NUMS            ; GET NUMERIC VALUE
1951  DA 0B1A         :       JP      C,ERRA          ; ARGUMENT ERROR
1954  EB              :AVAL   EX      DE,HL
1955  2A 78F2         :AVAL1  LD      HL,(OPRD)       ; FETCH OPERAND
1958  AF              :       XOR     A               ; GET A ZERO
1959  32 7AF2         :       LD      (OPRI),A        ; STORE IN OPERAND INDICATOR
                      :*  DETERMINE & PERFORM OPERATION (+,-,>,<,|,&,%,*,/)
195C  3A 77F2         :       LD      A,(OPER)        ; GET OPERATOR INDICATOR
195F  B7              :       OR      A               ; SET FLAGS
1960  282C            :       JR      Z,AADD-$
1962  FE21            :       CP      '!'             ; LOGICAL OR?
1964  2837            :       JR      Z,AOR-$
1966  FE25            :       CP      '%'             ; LOGICAL XOR?
1968  283B            :       JR      Z,AXOR-$
196A  FE26            :       CP      '&'             ; LOGICAL AND?
196C  283F            :       JR      Z,AAND-$
196E  FE2D            :       CP      '-'             ; SUBTRACT?
1970  2823            :       JR      Z,ASUB-$
1972  FE3E            :       CP      '>'             ; EXTRACT LOW?
1974  2813            :       JR      Z,ALOW-$
1976  FE3C            :       CP      '<'             ; EXTRACT HIGH
1978  280E            :       JR      Z,AHIGH-$
197A  FE2A            :       CP      '*'             ; MULTIPLY
197C  2805            :       JR      Z,AMULT-$
197E  CD 0BD1         :       CALL    DIV             ; HL=HL/DE
1981  180C            :       JR      ASC6-$
1983  CD 0BD1         :AMULT  CALL    MUL             ; HL=HL*DE
1986  1807            :       JR      ASC6-$
```

PROGRAM ERRORS -- 0

```
1988  6C              :AHIGH   LD      L,H             ; EXTRACT HIGH
1989  2600            :ALOW    LD      H,0             ; ZERO OUT HIGH PART
198B  19              :        ADD     HL,DE           ; ADD ARGUMENTS IF DUAL
198C  1801            :        JR      ASC6-$
198E  19              :AADD    ADD     HL,DE           ; PERFORM ADDITION OF ARGS
198F  22 78F2         :ASC6    LD      (OPRD),HL       ; SAVE RESULT
1992  C3 BB18         :        JP      NXT1
1995  7D              :ASUB    LD      A,L
1996  93              :        SUB     E
1997  6F              :        LD      L,A
1998  7C              :        LD      A,H
1999  9A              :        SBC     A,D
199A  67              :        LD      H,A
199B  18F2            :        JR      ASC6-$
199D  7B              :AOR     LD      A,E             ; LOGICAL OR
199E  B5              :        OR      L
199F  6F              :        LD      L,A
19A0  7A              :        LD      A,D
19A1  B4              :        OR      H
19A2  67              :        LD      H,A
19A3  18EA            :        JR      ASC6-$
19A5  7B              :AXOR    LD      A,E             ; LOGICAL EXCLUSIVE OR
19A6  AD              :        XOR     L
19A7  6F              :        LD      L,A
19A8  7A              :        LD      A,D
19A9  AC              :        XOR     H
19AA  67              :        LD      H,A
19AB  18E2            :        JR      ASC6-$
19AD  7B              :AAND    LD      A,E             ; LOGICAL AND
19AE  A5              :        AND     L
19AF  6F              :        LD      L,A
19B0  7A              :        LD      A,D
19B1  A4              :        AND     H
19B2  67              :        LD      H,A
19B3  18DA            :        JR      ASC6
19B5  CD 3D18         :ALAB    CALL    SLAB
19B8  289A            :        JR      Z,AVAL-$
19BA  384F            :        JR      C,ERRA-$        ; ILLEGAL SYMBOL
19BC  183D            :        JR      ERRU-$          ; UNDEFINED SYMBOL
                      :*
                      :* GET HERE WHEN TERMINATING CHAR IS FOUND.
                      :*  CHECK FOR LEADING FIELD SEPARATOR.
                      :*
19BE  3A 7AF2         :END     LD      A,(OPRI)        ; FETCH OPERAND INDICATOR
19C1  B7              :        OR      A               ; SET FLAGS
19C2  C2 F119         :        JP      NZ,ERRS         ; SYNTAX ERROR
19C5  2A 78F2         :        LD      HL,(OPRD)
19C8  7C              :GEN1    LD      A,H             ; GET HIGH ORDER BYTE
19C9  11 78F2         :        LD      DE,TEMP         ; GET ADDRESS
```

PROGRAM ERRORS -- 0

```
19CC  B7                :         OR      A               ; SET FLAGS
19CD  C9                :         RET
                        :*
                        :* GET A NUMERIC VALUE WHICH IS EITHER HEXADECIMAL OR
                        :*  DECIMAL.  ON RETURN, CARRY SET INDICATES AN ERROR.
                        :*
19CE  CD 8B18           :NUMS     CALL    ALPS            ; GET NUMERIC
19D1  1B                :         DEC     DE
19D2  1A                :         LD      A,(DE)          ; GET LAST CHAR
19D3  01 FDF1           :         LD      BC,ABUF         ; SET BUFFER ADDRESS
19D6  FE48              :         CP      'H'             ; IS IT HEXADECIMAL?
19D8  2809              :         JR      Z,NUM2-$
19DA  FE44              :         CP      'D'             ; IS IT DECIMAL?
19DC  2002              :         JR      NZ,NUM1-$
19DE  AF                :         XOR     A               ; GET A ZERO
19DF  12                :         LD      (DE),A          ; CLEAR D FROM BUFFER
19E0  C3 5602           :NUM1     JP      ADEC            ; CONVERT DECIMAL VALUE
19E3  AF                :NUM2     XOR     A               ; GET A ZERO
19E4  12                :         LD      (DE),A          ; CLEAR H FROM BUFFER
19E5  C3 6F02           :         JP      AHEX
                        :*  PROCESS REGISTER ERROR
19E8  3E52              :ERRR     LD      A,'R'           ; GET INDICATOR
19EA  21 0000           :ERRR1    LD      HL,0            ; GET A 0
19ED  32 ECFD           :         LD      (OBUF),A        ; SET IN OUTPUT BUFFER
19F0  C9                :         RET
                        :*  PROCESS SYNTAX ERROR
19F1  3E53              :ERRS     LD      A,'S'           ; GET INDICATOR
19F3  32 ECFD           :ERRS1    LD      (OBUF),A        ; STORE IN OUTPUT BUFFER
19F6  21 0000           :         LD      HL,0
19F9  18CD              :         JR      SEN1-$
                        :*  PROCESS UNDEFINED SYMBOL ERROR
19FB  3E55              :ERRU     LD      A,'U'           ; GET INDICATOR
19FD  18F4              :         JR      ERRS1-$
                        :*  PROCESS VALUE ERROR
19FF  3E56              :ERRV     LD      A,'V'           ; GET INDICATOR
1A01  18E7              :         JR      ERRR1-$
                        :*  PROCESS MISSING LABEL ERROR
1A03  3E4D              :ERRM     LD      A,'M'           ; GET INDICATOR
1A05  32 ECFD           :         LD      (OBUF),A        ; STORE IN OUTPUT BUFFER
1A08  C3 3514           :         JP      AOU1
                        :*  PROCESS ARGUMENT ERROR
1A0B  3E41              :ERRA     LD      A,'A'           ; GET INDICATOR
1A0D  18E4              :         JR      ERRS1-$
                        :*
                        :* PROCESS OPCODE ERROR ---
                        :*  STORE 3 BYTES OF ZERO IN OBJECT CODE TO PROVIDE
                        :*  FOR A PATCH.
                        :*
1A0F  3E4F              :ERRO     LD      A,'O'           ; GET INDICATOR

PROGRAM ERRORS --      0
```

```
1A11  32 ECFD            :ERRO1   LD      (OBUF),A        ; STORE IN OUTPUT BUFFER
1A14  3A 71F2            :        LD      A,(PASS)        ; FETCH PASS INDICATOR
1A17  B7                 :        OR      A               ; WHICH PASS
1A18  C8                 :        RET     Z               ; RETURN IF PASS 1
1A19  0E03               :        LD      C,3             ; NEED 3 BYTES
1A1B  AF                 :ERRO2   XOR     A               ; GET A ZERO
1A1C  CD 1717            :        CALL    ASTO            ; PUT IN LISTING AND MEMORY
1A1F  0D                 :        DEC     C
1A20  20F9               :        JR      NZ,ERRO2-$
1A22  C9                 :        RET
                         :* PROCESS LABEL ERROR
1A23  3E4C               :ERRL    LD      A,'L'           ; GET INDICATOR
1A25  18EA               :        JR      ERRO1-$
                         :* PROCESS DUPLICATE LABEL ERROR
1A27  3E44               :ERRD    LD      A,'D'           ; GET ERROR INDICATOR
1A29  32 ECFD            :        LD      (OBUF),A        ; STORE IN OUTPUT BUFFER
1A2C  CD 1214            :        CALL    AOUT            ; DISPLAY ERROR
1A2F  C3 9B14            :        JP      OPC             ; PROCESS OPCODE

PROGRAM ERRORS --      0
```

```
                        :**** COMMAND -- BREK
                        :*
                        :*
                        :* THIS ROUTINE SETS OR CLEARS BREAKPOINTS.
                        :*
1A32                    :BREAK  EQU     $
1A32  FE4C              :       CP      'L'             ; LIST IF L
1A34  CA BF1A           :       JP      Z,BREKE
1A37  FE44              :       CP      'D'             ; DELETE BP
1A39  CA 751A           :       JP      Z,BRKPD
1A3C  FE53              :       CP      'S'             ; SCRATCH (CLEAR) ALL BREAKPOINTS?
1A3E  CA A61A           :       JP      Z,CLRB          ; DO IT
1A41  CD A502           :       CALL    CKA1            ; MUST BE AT LEAST ONE ARG
1A44  1608              :       LD      D,NBR           ; ELSE, GET NUMBER OF BREAKPOINTS
1A46  21 A9F1           :       LD      HL,BRT          ; AND ADDR OF TABLE
1A49  7E                :B1     LD      A,(HL)          ; GET HI BYTE OF ENTRY
1A4A  23                :       INC     HL
1A4B  46                :       LD      B,(HL)          ; GET LO BYTE OF ENTRY
1A4C  B0                :       OR      B               ; CHECK FOR EMPTY ENTRY
1A4D  2808              :       JR      Z,B2-$          ; BRANCH IF EMPTY
1A4F  23                :       INC     HL              ; ELSE GO ON TO NEXT ENTRY
1A50  23                :       INC     HL
1A51  15                :       DEC     D               ; BUMP COUNT
1A52  20F5              :       JR      NZ,B1-$         ; AND TRY AGAIN
1A54  C3 2605           :       JP      CUSE1           ; BP TABLE FULL
1A57  2B                :B2     DEC     HL
1A58  ED5B 0DF2         :       LD      DE,(BBUF)       ; GET ADDRESS
1A5C  7A                :       LD      A,D             ; CHECK FOR ADDR > 11D
1A5D  B7                :       OR      A
1A5E  200B              :       JR      NZ,B3-$
1A60  7B                :       LD      A,E
1A61  FE0B              :       CP      11
1A63  3006              :       JR      NC,B3-$
1A65  21 811C           :IBPMS  LD      HL,MS92         ; 'INVALID BP'
1A68  C3 6603           :       JP      MESS
1A6B  72                :B3     LD      (HL),D          ; SAVE ADDRESS
1A6C  23                :       INC     HL
1A6D  73                :       LD      (HL),E
1A6E  23                :       INC     HL
1A6F  1A                :       LD      A,(DE)          ; PICK UP INSTRUCTION
1A70  77                :       LD      (HL),A          ; SAVE IT
1A71  3ECF              :       LD      A,0CFH          ; REPLACE IT WITH A
1A73  12                :       LD      (DE),A          ; RESTART INSTRUCTION
1A74  C9                :       RET                     ; THEN RETURN
1A75  CD A502           :BRKPD  CALL    CKA1            ; CHECK FOR ARG
1A78  2A D7F1           :       LD      HL,(PCSAV)      ; SAVE PC TEMPORRAILY
1A7B  22 0FF2           :       LD      (BBUF+2),HL
1A7E  2A 0DF2           :       LD      HL,(BBUF)       ; GET ARG
1A81  22 D7F1           :       LD      (PCSAV),HL      ; SET PCADR

PROGRAM ERRORS --       0
```

```
1A84  EB              :        EX      DE,HL           ; IN DE
1A85  CD E01A         :        CALL    BRKPE
1A88  2A 0FF2         :        LD      HL,(BBUF+2)     ; RESTORE PCSAV
1A8B  22 D7F1         :        LD      (PCSAV),HL
1A8E  C9              :        RET
                      :*
                      :*  THIS ROUTINE EXAMINES AND DISPLAYS ALL BP ADDRESSES
                      :*
1A8F                  :BREKE   EQU     $
1A8F  21 A9F1         :        LD      HL,BRT          ; POINT TO TABLE
1A92  0E08            :        LD      C,NBR           ; NUMBER OF BPS
1A94  CD 3FD0         :        CALL    CRLF
1A97  7E              :BRKEL   LD      A,(HL)          ; GET BYTE
1A98  23              :        INC     HL              ; POINT TO NEXT
1A99  B6              :        OR      (HL)            ; ZERO ADDRESS?
1A9A  2804            :        JR      Z,BRKEZ-$
1A9C  2B              :        DEC     HL              ; GET BP ADR
1A9D  CD 5003         :        CALL    PDEADR          ; PRINT BP ADR
1AA0  23              :BRKEZ   INC     HL              ; SKIP REPLACED BYTE
1AA1  23              :        INC     HL              ; POINT TO NEXT ADR
1AA2  0D              :        DEC     C               ; DECR COUNT
1AA3  20F2            :        JR      NZ,BRKEL-$
1AA5  C9              :        RET                     ; DONE
                      :*
                      :* THIS ROUTINE CLEARS ALL BREAKPOINTS.
                      :*
1AA6                  :CLRB    EQU     $
1AA6  21 A9F1         :        LD      HL,BRT          ; GET TABLE ADDRESS
1AA9  0608            :        LD      B,NBR           ; GET NUMBER OF BREAKPOINTS
1AAB  AF              :CLBL    XOR     A               ; GET A ZERO
1AAC  56              :        LD      D,(HL)          ; GET HI BYTE OF ENTRY
1AAD  77              :        LD      (HL),A
1AAE  23              :        INC     HL
1AAF  5E              :        LD      E,(HL)          ; GET LO BYTE OF ENTRY
1AB0  77              :        LD      (HL),A
1AB1  23              :        INC     HL
1AB2  4E              :        LD      C,(HL)          ; GET INST BYTE
1AB3  23              :        INC     HL
1AB4  7A              :        LD      A,D             ; WAS THIS A NULL ENTRY?
1AB5  B3              :        OR      E
1AB6  2802            :        JR      Z,CL2-$         ; BRANCH IF IT WAS
1AB8  79              :        LD      A,C
1AB9  12              :        LD      (DE),A          ; ELSE, PLUG INST BACK IN
1ABA  10EF            :CL2     DJNZ    CLBL-$          ; GO DO NEXT ONE
1ABC  C9              :        RET                     ; RETURN WHEN DONE
                      :*
                      :* COME HERE WHEN WE HIT A BREAKPOINT.
                      :*
1ABD                  :BRKP    EQU     $

PROGRAM ERRORS --     0
```

```
1ABD  22 D3F1            :        LD      (HLSAV),HL      ; SAVE HL
1AC0  E1                 :        POP     HL
1AC1  2B                 :        DEC     HL
1AC2  22 D7F1            :        LD      (PCSAV),HL      ; SAVE PC
1AC5  ED73 D5F1          :        LD      (SPSAV),SP
1AC9  31 D3F1            :        LD      SP,HLSAV
1ACC  D5                 :        PUSH    DE              ; SAVE DE
1ACD  C5                 :        PUSH    BC              ; SAVE BC
1ACE  F5                 :        PUSH    AF              ; SAVE AF
1ACF  DDE5               :        PUSH    IX              ; SAVE IX
1AD1  FDE5               :        PUSH    IY              ; SAVE IY
1AD3  08                 :        EX      AF,AF'          ; GET ALTERNATE REGISTER SET
1AD4  D9                 :        EXX
1AD5  E5                 :        PUSH    HL              ; SAVE HLP
1AD6  D5                 :        PUSH    DE              ; SAVE DEP
1AD7  C5                 :        PUSH    BC              ; SAVE BCP
1AD8  F5                 :        PUSH    AF              ; SAVE AFP
1AD9  31 00F3            :        LD      SP,STACK        ; SET SP AGAIN
1ADC  ED5B D7F1          :        LD      DE,(PCSAV)      ; GET PC
1AE0  21 A9F1            :BRKPE   LD      HL,BRT          ; GET ADDR OF TABLE
1AE3  0608               :        LD      B,NBR           ; AND NUMBER OF ENTRIES
1AE5  7E                 :BL1     LD      A,(HL)          ; GET AN ENTRY FROM THE TABLE
1AE6  23                 :        INC     HL
1AE7  BA                 :        CP      D               ; DOES IT MATCH
1AE8  2004               :        JR      NZ,BL2-$        ; BRANCH IF NOT
1AEA  7E                 :        LD      A,(HL)          ; ELSE GET NEXT BYTE
1AEB  BB                 :        CP      E               ; CHECK IT
1AEC  2807               :        JR      Z,BL3-$         ; IT MATCHES
1AEE  23                 :BL2     INC     HL              ; BUMP AROUND THIS ENTRY
1AEF  23                 :        INC     HL
1AF0  10F3               :        DJNZ    BL1-$           ; BUMP COUNT
1AF2  C3 651A            :        JP      IBPMS           ; 'INVALID BP'
1AF5  23                 :BL3     INC     HL
1AF6  7E                 :        LD      A,(HL)          ; GET INSTR BYTE
1AF7  12                 :        LD      (DE),A          ; PUT IT BACK
1AF8  AF                 :        XOR     A               ; CLEAR ENTRY IN TABLE
1AF9  2B                 :        DEC     HL
1AFA  77                 :        LD      (HL),A
1AFB  2B                 :        DEC     HL
1AFC  77                 :        LD      (HL),A
1AFD  21 9E1B            :        LD      HL,MS13         ; REGISTER MESSAGE
1B00  0E06               :        LD      C,6             ; NUMBER OF REGS IN FIRST LINE
1B02  11 D8F1            :        LD      DE,PCSAV+1      ; POINT TO TOP OF REG AREA
1B05  CD 101B            :        CALL    PRLINE          ; PRINT REG LINE
1B08  0E06               :        LD      C,6             ; PRINT SIX REGS ON SECOND LINE
1B0A  CD 101B            :        CALL    PRLINE          ; PRINT
1B0D  C3 BA00            :        JP      EOR
                         ;*  PRINT LINE OF REGISTER INFORMATION
1B10  CD 3ED0            :PRLINE  CALL    CRLF            ; PRINT REG LINE
```

PROGRAM ERRORS -- 0

```
1813  7E              :PRL1   LD      A,(HL)          ; PRINT REG NAME
1814  CD 1ED0         :       CALL    OUTPUT
1817  23              :       INC     HL
1818  7E              :       LD      A,(HL)
1819  CD 1ED0         :       CALL    OUTPUT
181C  23              :       INC     HL
181D  7E              :       LD      A,(HL)
181E  CD 1ED0         :       CALL    OUTPUT
1821  23              :       INC     HL
1822  CD 42D0         :       CALL    BLK1            ; PRINT BLANK
1825  1A              :       LD      A,(DE)          ; PRINT VALUE
1826  CD 60D0         :       CALL    HOUT
1829  1B              :       DEC     DE
182A  1A              :       LD      A,(DE)          ; 2ND BYTE
182B  CD 60D0         :       CALL    HOUT
182E  1B              :       DEC     DE
182F  CD 42D0         :       CALL    BLK1            ; 2 BLANKS
1832  CD 42D0         :       CALL    BLK1
1835  0D              :       DEC     C               ; DECR LINE ELEMENT COUNT
1836  20DB            :       JR      NZ,PRL1-$
1838  C9              :       RET

PROGRAM ERRORS --     0
```

```
                        :**** COMMAND -- CONT
                        :*
                        :* THIS ROUTINE PROCEEDS FROM A BREAKPOINT
                        :*
1839                    :CONT   EQU     $
1839  CD 3FD0           :       CALL    CRLF
183C  CD AF02           :       CALL    CKA11           ; CHECK FOR ARG
183F  2806              :       JR      Z,P1-$          ; JUMP IF NO ARG
1841  2A 0DF2           :       LD      HL,(BBUF)       ; ELSE, GET ARG
1844  22 D7F1           :       LD      (PCSAV),HL      ; PLUG IT INTO PC SLOT
1847  31 C1F1           :P1     LD      SP,REGSAV       ; SET SP TO POINT AT REG'S
                        :*  RESTORE REGISTERS FROM REGISTER SAVE AREA
184A  F1                :       POP     AF
184B  C1                :       POP     BC
184C  D1                :       POP     DE
184D  E1                :       POP     HL
184E  08                :       EX      AF,AF'          ; STORE ALTERNATE REGISTER SET
184F  D9                :       EXX
1850  FDE1              :       POP     IY              ; GET IY
1852  DDE1              :       POP     IX              ; GET IX
1854  F1                :       POP     AF              ; GET AF
1855  C1                :       POP     BC              ; GET BC
1856  D1                :       POP     DE              ; GET DE
1857  E1                :       POP     HL              ; GET SP
1858  E1                :       POP     HL
1859  F9                :       LD      SP,HL           ; RESTORE OLD SP
185A  2A D7F1           :       LD      HL,(PCSAV)      ; GET PC
185D  E5                :       PUSH    HL              ; PUT IT ON STACK
185E  2A D3F1           :       LD      HL,(HLSAV)      ; RESTORE H,L
1861  C9                :       RET                     ; AND PROCEED

PROGRAM ERRORS --       0
```

:********* **:*******
:********* ARIAN SYSTEM TABLES **:*******
:********* **:*******

PROGRAM ERRORS -- 0

```
                        ;********
                        ;******** SYSTEM MESSAGE BLOCK ********
                        ;********
                        ;
1B62  2A2A2041524941    ;MS0    DEFM    '** ARIAN II **',0DH
      4E204949202A2A
      0D
1B71  4E4F20464E0D      ;MS1    DEFM    'NO FN',0DH
1B77  4E4F2041524700D   ;MS2    DEFM    'NO ARG',0DH
1B7E  4E4F20324E4420    ;MS3    DEFM    'NO 2ND ARG',0DH
      41524700
1B89  504152414D2045    ;MS4    DEFM    'PARAM ERR',0DH
      52520D
1B93  494E564C442043    ;MS5    DEFM    'INVLD CMND',0DH
      4D4E440D
1B9E  504320535020048   ;MS13   DEFM    'PC SP HL DE BC AF IX IY HL',27H,'DE',27H,'BC',27H,'AF',27H
      4C20445204243
      20414620495820
      495920484C2744
      4527424327414627
1BC2  46494C45205459    ;MS22   DEFM    'FILE TYPE ERR',0DH
      50452045525200
1BD0  56414C49142046    ;MS25   DEFM    'VALID FILE',0DH
      494C450D
1BDB  494E564C442046    ;MS26   DEFM    'INVLD FILE',0DH
      494C450D
1BE6  494E564C442043    ;MS27   DEFM    'INVLD CDRIVE',0DH
      44524956450D
1BF3  4E4F2046494C45    ;MS31   DEFM    'NO FILE',0DH
      0D
1BFB  5245504C414345    ;MS32   DEFM    'REPLACE',0DH
      0D
1C03  46494C45205341    ;MS33   DEFM    'FILE SAVED',0DH
      5645440D
1C0E  4E4557204E414D    ;MS35   DEFM    'NEW NAME? ',0DH
      453F200D
1C19  46494C45204E4F    ;MS36   DEFM    'FILE NOT FOUND',0DH
      5420464F554E44
      0D
1C28  4449522F544142    ;MS38   DEFM    'DIR/TABLE FULL',0DH
      4C452046554C4C
      0D
1C37  44555020464E0D    ;MS39   DEFM    'DUP FN',0DH
1C3E  4449534B204552    ;MS40   DEFM    'DISK ERR',0DH
      520D
1C47  4C4E554D204F56    ;MS50   DEFM    'LNUM OVFL',0DH
      464C0D
1C51  52414E47452045    ;MS51   DEFM    'RANGE ERR',0DH

PROGRAM ERRORS --      0
```

```
                52520D
1C5B  41534D20504153    :MS60   DEFM    'ASM PASS 1',0DH
                5320310D
1C66  41534D20504153    :MS61   DEFM    'ASM PASS 2',0DH
                5320320D
1C71  455845432052      :MS72   DEFM    'EXEC TRAP',0DH
                41500D
1C7B  2420454F4C0D      :MS80   DEFM    '$ EOL',0DH
1C81  494E564C  2042    :MS92   DEFM    'INVLD BP',0DH
                500D
1C8A  575320455252 0D   :MS93   DEFM    'WS ERR',0DH
1C91  2E434D44          :EXT    DB      '.CMD'          ; EXTENSION OF CL3 COMMANDS
1C95  39393939          :LMAX   DB      '9999'          ; MAXIMUM LINE NUMBER

PROGRAM ERRORS --       0
```

```
                        ;********
                        ;******** LEGAL COMMAND TABLE ********
                        ;********
                        ;
0021                    ;NCOM    EQU     33              ; NUMBER OF COMMANDS
1C99  5554494C          ;CTAB    DEFM    'UTIL'          ; UTIL COMMAND
1C9D  0806              ;        DEFW    UTIL
1C9F  45584543          ;        DEFM    'EXEC'          ; EXECUTE COMMAND
1CA3  8003              ;        DEFW    EXEC
1CA5  46494C45          ;        DEFM    'FILE'          ; FILE COMMAND
1CA9  5C0D              ;        DEFW    FILE
1CAB  4C495354          ;        DEFM    'LIST'          ; LIST COMMAND
1CAF  2210              ;        DEFW    LIST
1CB1  44454C20          ;        DEFM    'DEL '          ; DELETE COMMAND
1CB5  8711              ;        DEFW    DELL
1CB7  41535340          ;        DEFM    'ASSM'          ; ASSEMBLE COMMAND
1CBB  5213              ;        DEFW    ASSM
1CBD  43555354          ;        DEFM    'CUST'          ; CUSTOMER COMMAND
1CC1  2304              ;        DEFW    CUST
1CC3  42524548          ;        DEFM    'BREK'          ; BREAKPOINT COMMAND
1CC7  321A              ;        DEFW    BREAK
1CC9  434F4E54          ;        DEFM    'CONT'          ; CONTINUE COMMAND
1CCD  3918              ;        DEFW    CONT
1CCF  524E5540          ;        DEFM    'RNUM'          ; RENUMBER COMMAND
1CD3  2511              ;        DEFW    RNUM
1CD5  41504E44          ;        DEFM    'APND'          ; AUTOMATIC LINE NUMBERING
1CD9  F00B              ;        DEFW    AUTO
1CDB  4C4E414D          ;        DEFM    'LNAM'          ; LOCAL FILE RENAME COMMAND
1CDF  1B0C              ;        DEFW    RNME
1CE1  444E414D          ;        DEFM    'DNAM'          ; DISK FILE RENAME COMMAND
1CE5  6B09              ;        DEFW    DNAME
1CE7  44444952          ;        DEFM    'DDIR'          ; DISK DIRECTORY
1CEB  690C              ;        DEFW    DIRD
1CED  4C444952          ;        DEFM    'LDIR'          ; MEM FILE DIRECTORY
1CF1  4C0C              ;        DEFW    LDIR
1CF3  4C534352          ;        DEFM    'LSCR'          ; SCRATCH FILES COMMAND
1CF7  4705              ;        DEFW    SCR
1CF9  52435652          ;        DEFM    'RCVR'          ; RECOVER FILE COMMAND
1CFD  F905              ;        DEFW    RCVR
1CFF  4C44454C          ;        DEFM    'LDEL'          ; FILE DELETE COMMAND
1D03  EE0C              ;        DEFW    FDEL
1D05  5445524D          ;        DEFM    'TERM'          ; LINK COMMAND
1D09  1D04              ;        DEFW    LINK
1D0B  45444954          ;        DEFM    'EDIT'          ; LINE EDIT COMMAND
1D0F  8806              ;        DEFW    LEDIT
1D11  4C4F4144          ;        DEFM    'LOAD'          ; FILE LOAD COMMAND
1D15  4308              ;        DEFW    LOAD
1D17  45584954          ;        DEFM    'EXIT'          ; EXIT TO FDOS
1D1B  5405              ;        DEFW    EXIT

PROGRAM ERRORS --       0
```

```
1D1D  53594D54          :       DEFM    'SYMT'          ; SYMBOL TABLE COMMAND
1D21  F811              :       DEFW    SYMB
1D23  53415645          :       DEFM    'SAVE'          ; FILE SAVE COMMAND
1D27  7A09              :       DEFW    DSAVE
1D29  4444454C          :       DEFM    'DDEL'          ; DELETE DISK FILE
1D2D  5809              :       DEFW    DELDF
1D2F  46434848          :       DEFM    'FCHK'          ; FILE CHECK
1D33  0A09              :       DEFW    FCHK
1D35  49535254          :       DEFM    'ISRT'          ; INSERT COMMAND
1D39  510B              :       DEFW    INS
1D3B  434D4E44          :       DEFM    'CMND'          ; INVOKE LEVEL 3 COMMANDS
1D3F  5F05              :       DEFW    CMND
1D41  54414253          :       DEFM    'TABS'          ; TAB SET COMMAND
1D45  4F0D              :       DEFW    TABST
1D47  52455345          :       DEFM    'RESE'          ; RESET COMMAND
1D4B  9A00              :       DEFW    INITA
1D4D  46494E44          :       DEFM    'FIND'          ; STRING FIND COMMAND
1D51  5506              :       DEFW    FINDS
1D53  5052494E          :       DEFM    'PRIN'          ; PRINT COMMAND
1D57  1D10              :       DEFW    LISTP
1D59  53455443          :       DEFM    'SETC'          ; SET/RESET REDIRECTABLE I/O
1D5D  4206              :       DEFW    SETC
                        :*
                        :*  *** END OF COMMAND TABLE ***
                        :*
```

PROGRAM ERRORS -- 0

```
                        ;********                                  ********
                        ;******** SYSTEM SYMBOL TABLE (SSYMT) ********
                        ;********                                  ********
                        ;
                        ;
0019                    NSYM    EQU     25              ; NUMBER OF SYSTEM SYMBOLS
1D5F  C8                ;       DEFB    NSYM*8          ; NUMBER OF BYTES TAKEN UP BY TABLE
1D60  19                SSYMT   DEFB    NSYM            ; NUMBER OF SYMBOLS
1D61  50535700          ;       DEFM    'PSW',0         ; SET PSW VALUE
1D65  0600              ;       DEFW    6
1D67  53500000          ;       DEFM    'SP',0,0        ; SET SP VALUE
1D6B  0600              ;       DEFW    6
1D6D  5A154F52          ;       DEFM    'ZEOR'          ; EOR
1D71  6600              ;       DEFW    XEOR
1D73  5A4C494E          ;       DEFM    'ZLIN'          ; READ INPUT LINE ROUTINE
1D77  6F00              ;       DEFW    XREAD
1D79  5A494E4B          ;       DEFM    'ZINK'          ; INK (ABORT IF <ESC> IS TYPED BY USER)
1D7D  7200              ;       DEFW    XINK
1D7F  5A494E00          ;       DEFM    'ZIN',0         ; INPUT
1D83  1BD0              ;       DEFW    INPUT
1D85  5A4F5554          ;       DEFM    'ZOUT'          ; OUTPUT
1D89  1ED0              ;       DEFW    OUTPUT
1D8B  5A435200          ;       DEFM    'ZCR',0         ; CRLF
1D8F  3FD0              ;       DEFW    CRLF
1D91  5A415247          ;       DEFM    'ZARG'          ; ARGUMENT PARSER
1D95  6C00              ;       DEFW    XPARSE
1D97  5A484F54          ;       DEFM    'ZHOT'          ; PRINT A AS HEX
1D9B  60D0              ;       DEFW    HOUT
1D9D  5A444F54          ;       DEFM    'ZDOT'          ; PRINT A AS DEC
1DA1  66D0              ;       DEFW    DOUT
1DA3  5A424C48          ;       DEFM    'ZBLK'          ; PRINT A <SP>
1DA7  42D0              ;       DEFW    BLK1
1DA9  5A505248          ;       DEFM    'ZPRH'          ; PRINT LINE ADR BY H&L AND ENDING IN <CR>
1DAD  51D0              ;       DEFW    SCRNA
1DAF  5A505048          ;       DEFM    'ZPPH'          ; SAME AS ZPRH, EXCEPT PRINTS ON PRINTER
1DB3  4FD0              ;       DEFW    PPRINT
1DB5  5A505252          ;       DEFM    'ZPRR'          ; PRINT LINE ADR BY RET ADR AND ENDING IN 0
1DB9  4BD0              ;       DEFW    PRINT
1DBB  5A50484C          ;       DEFM    'ZPHL'          ; PRINT H&L IN HEX FOLLOWED BY A BLANK
1DBF  6FD0              ;       DEFW    PHLB
1DC1  5A424F46          ;       DEFM    'ZBOF'          ; BEGINNING OF FILE POINTER
1DC5  98F0              ;       DEFW    BOFP
1DC7  5A454F46          ;       DEFM    'ZEOF'          ; END OF FILE POINTER
1DCB  9AF0              ;       DEFW    EOFP
1DCD  5A424246          ;       DEFM    'ZBBF'          ; BINARY BUFFER POINTER
1DD1  0DF2              ;       DEFW    BBUF
1DD3  5A494246          ;       DEFM    'ZIBF'          ; INPUT BUFFER POINTER
1DD7  01FE              ;       DEFW    IBUF
1DD9  5A434841          ;       DEFM    'ZCHA'          ; CONVERT HEX TO ASCII

PROGRAM ERRORS --       0
```

```
1DDD   93D0             :       DEFW    BINI
1DDF   5A434148         :       DEFM    'ZCAH'          ; CONVERT ASCII TO HEX
1DE3   96D0             :       DEFW    AHSI
1DE5   5A454E00         :       DEFM    'ZEN',0         ; EXCHANGE NYBBLES IN A
1DE9   99D0             :       DEFW    EN
1DEB   5A4A524C         :       DEFM    'ZJRL'          ; JUMP RELATIVE LONG
1DEF   BAD0             :       DEFW    JREL
1DF1   5A43524C         :       DEFM    'ZCRL'          ; CALL RELATIVE LONG
1DF5   B7D0             :       DEFW    CREL

PROGRAM ERRORS --       0
```

```
                        :********                    ********
                        :******** OPCODE TABLE ********
                        :********                    ********
                        :
                        :
1DF7  4F52470000        :OTAB    DB      'ORG',0,0
1DFC  4551550001        :        DB      'EQU',0,1
1E01  4442000002        :        DB      'DB',0,0,2
1E06  4453000003        :        DB      'DS',0,0,3
1E0B  4457000005        :        DB      'DW',0,0,5
1E10  4153430007        :        DB      'ASC',0,7
1E15  454E440006        :        DB      'END',0,6
1E1A  4C53540008        :        DB      'LST',0,8
1E1F  4E4C535409        :        DB      'NLST',9
1E24  455845430A        :        DB      'EXEC',10
                        :**  Z80 4-BYTE OPCODES **
1E29  5353504473        :        DB      'SSPD',115      ; LD (NN),SP
1E2E  4C5350447B        :        DB      'LSPD',123      ; LD SP,(NN)
1E33  5342434443        :        DB      'SBCD',67       ; LD (NN),BC
1E38  4C42434448        :        DB      'LBCD',75       ; LD BC,(NN)
1E3D  5344454453        :        DB      'SDED',83       ; LD (NN),DE
1E42  4C4445445B        :        DB      'LDED',91       ; LD DE,(NN)
                        :** END OF Z80 4-BYTE OPCODES **
1E47  00                :        DB      0               ; END OF SECTION
1E48  45584108          :        DB      'EXA',8         ** Z80 EX AF,AF'
1E4C  45585BD9          :        DB      'EXX',217       ** Z80 EXX
1E50  484C5476          :        DB      'HLT',118
1E54  524C4307          :        DB      'RLC',7
1E58  5252430F          :        DB      'RRC',15
1E5C  52414C17          :        DB      'RAL',23
1E60  5241521F          :        DB      'RAR',31
1E64  524554C9          :        DB      'RET',201
1E68  434D412F          :        DB      'CMA',47
1E6C  53544337          :        DB      'STC',55
1E70  44414127          :        DB      'DAA',39
1E74  434D433F          :        DB      'CMC',63
1E78  454900FB          :        DB      'EI',0,251
1E7C  444900F3          :        DB      'DI',0,243
1E80  4E4F5000          :        DB      'NOP',0
1E84  00                :        DB      0               ; END OF SECTION
1E85  58434847EB        :        DB      'XCHG',235
1E8A  5854484CE3        :        DB      'XTHL',227
1E8F  5350484CF9        :        DB      'SPHL',249
1E94  5043484CE9        :        DB      'PCHL',233
1E99  00                :        DB      0               ; END OF SECTION
1E9A  5354415802        :        DB      'STAX',2
1E9F  4C4441580A        :        DB      'LDAX',10
1EA4  00                :        DB      0               ; END OF SECTION
1EA5  5055534BC5        :        DB      'PUSH',197

PROGRAM ERRORS --    0
```

```
1EAA 504F5000C1          :          DB      'POP',0,193
1EAF 494E580003          :          DB      'INX',0,3
1EB4 444358000B          :          DB      'DCX',0,11
1EB9 4441440009          :          DB      'DAD',0,9
1EBE 00                  :          DB      0                   ; END OF SECTION
1EBF 494E5204            :          DB      'INR',4
1EC3 44435205            :          DB      'DCR',5
1EC7 4D4F5640            :          DB      'MOV',64
1ECB 41444480            :          DB      'ADD',128
1ECF 41444388            :          DB      'ADC',136
1ED3 53554290            :          DB      'SUB',144
1ED7 53424298            :          DB      'SBB',152
1EDB 414E41A0            :          DB      'ANA',160
1EDF 585241A8            :          DB      'XRA',168
1EE3 4F5241B0            :          DB      'ORA',176
1EE7 434D50B8            :          DB      'CMP',184
1EEB 525354C7            :          DB      'RST',199
1EEF 00                  :          DB      0                   ; END OF SECTION
                         :** Z80 BRANCH RELATIVE SECTION **
1EF0 42520018            :          DB      'BR',0,24           ; JR
1EF4 42430038            :          DB      'BC',0,56           ; JR C,X
1EF8 425A0028            :          DB      'BZ',0,40           ; JR Z,X
1EFC 424E4330            :          DB      'BNC',48            ; JR NC,X
1F00 424E5A20            :          DB      'BNZ',32            ; JR NZ,X
1F04 44424A10            :          DB      'DBJ',16            ; DJNZ
                         :** Z80 SPECIAL 2-BYTE OPCODE SECTION **
1F08 4C4400A8            :          DB      'LD',0,168          ; LDD
1F0C 4C4452B8            :          DB      'LDR',184           ; LDDR
1F10 4C4900A0            :          DB      'LI',0,160          ; LDI
1F14 4C4952B0            :          DB      'LIR',176           ; LDIR
1F18 43440A9             :          DB      'CD',0,169          ; CPD
1F1C 434452B9            :          DB      'CDR',185           ; CPDR
1F20 434900A1            :          DB      'CI',0,161          ; CPI
1F24 4349 52B1           :          DB      'CIR',177           ; CPIR
1F28 4E454744            :          DB      'NEG',68            ; NEG
1F2C 524C446F            :          DB      'RLD',111           ; RLD
1F30 52524467            :          DB      'RRD',103           ; RRD
1F34 53424242            :          DB      'SBB',66            ; SBC HL,BC
1F38 53424452            :          DB      'SBD',82            ; SBC HL,DE
1F3C 53485372            :          DB      'SHS',114           ; SBC HL,SP
1F40 4148424A            :          DB      'AHB',74            ; ADC HL,BC
1F44 4148445A            :          DB      'AHD',90            ; ADC HL,DE
1F48 4148537A            :          DB      'AHS',122           ; ADC HL,SP
1F4C 494D3046            :          DB      'IM0',70            ; IM 0
1F50 494D3156            :          DB      'IM1',86            ; IM 1
1F54 494D325E            :          DB      'IM2',94            ; IM 2
1F58 494400AA            :          DB      'ID',0,170          ; IND
1F5C 494452BA            :          DB      'IDR',186           ; INDR
1F60 494900A2            :          DB      'II',0,162          ; INI

PROGRAM ERRORS --      0
```

```
1F64  49495282          :       DB      'IIR',178       ; INIR
1F68  4F4400AB          :       DB      'OD',0,171      ; OUTD
1F6C  4F44528B          :       DB      'ODR',139       ; OTDR
1F70  4F4900A3          :       DB      'OI',0,163      ; OUTI
1F74  4F495283          :       DB      'OIR',179       ; OTIR
1F78  43494E78          :       DB      'CIN',120       ; IN A,(C)
1F7C  434F5479          :       DB      'COT',121       ; OUT (C),A
                        :** END OF Z80 2-BYTE SECTION **
1F80  414449C6          :       DB      'ADI',198
1F84  414349CE          :       DB      'ACI',206
1F88  535549D6          :       DB      'SUI',214
1F8C  534249DE          :       DB      'SBI',222
1F90  414E49E6          :       DB      'ANI',230
1F94  584249EE          :       DB      'XRI',238
1F98  4F5249F6          :       DB      'ORI',246
1F9C  435049FE          :       DB      'CPI',254
1FA0  494E00DB          :       DB      'IN',0,219
1FA4  4F5554D3          :       DB      'OUT',211
1FA8  4D564906          :       DB      'MVI',6
1FAC  00                :       DB      0               ; END OF SECTION
1FAD  4A4D5000C3        :       DB      'JMP',0,195
1FB2  43414C4CCD        :       DB      'CALL',205
1FB7  4C58490001        :       DB      'LXI',0,1
1FBC  4C4441003A        :       DB      'LDA',0,58
1FC1  5354410032        :       DB      'STA',0,50
1FC6  53484C4422        :       DB      'SHLD',34
1FCB  4C484C442A        :       DB      'LHLD',42
1FD0  00                :       DB      0               ; END OF SECTION
                        :*  CONDITION CODE TABLE
1FD1  4E5A00            :       DB      'NZ',0
1FD4  5A0008            :       DB      'Z',0,8
1FD7  4E4310            :       DB      'NC',16
1FDA  430018            :       DB      'C',0,24
1FDD  504F20            :       DB      'PO',32
1FE0  504528            :       DB      'PE',40
1FE3  500030            :       DB      'P',0,48
1FE6  4D0038            :       DB      'M',0,56
1FE9  00                :       DB      0               ; END OF SECTION
                        :*
                        :* PREDEFINE REGISTER VALUES IN THIS TABLE
                        :*
1FEA  4107              :RTAB   DB      'A',7
1FEC  4200              :       DB      'B',0
1FEE  4301              :       DB      'C',1
1FF0  4402              :       DB      'D',2
1FF2  4503              :       DB      'E',3
1FF4  4804              :       DB      'H',4
1FF6  4C05              :       DB      'L',5
1FF8  4D06              :       DB      'M',6

PROGRAM ERRORS --       0
```

```
1FFA  5006                :         DB      'P',6
1FFC  5306                :         DB      'S',6
1FFE  00                  :         DB      0               ; END OF TABLE INDICATOR
1FFE                      :ENDALL   EQU     $-1

PROGRAM ERRORS --     0
```

```
                        :********                                ********
                        :******** ARIAN SYSTEM RAM BUFFERS ********
                        :********                                ********
                        :
                        :
                        :         ORG     0F000H
                        :*
                        :*  WORKSPACE BUFFERS
                        :*
F000                    :WSPS     DS      2               ; WORKSPACE PTR START
F002                    :WSPE     DS      2               ; WORKSPACE PTR END
F004                    :WSBF     DS      2               ; WORKSPACE BUFFER
                        :*  LINK ADDR
D000                    :LINK1    EQU     0D000H
                        :*  INSERT PARAMETERS
F006                    :INSNL    DS      2
F008                    :INSLL    DS      2
                        :*  MEMORY MANAGEMENT PTRS
F006                    :FCUR     EQU     INSNL
F008                    :FCURE    EQU     INSLL
                        :*  DISK DRIVE NUMBER
F00A                    :DRIVEN   DS      1
                        :*  EXECUTION ADDRESS
F00B                    :EXADR    DS      2
                        :*  COMMAND LEVEL 3 EXECUTION ADDRESS
F00D                    :CEXADR   DS      2
                        :*  DISK LOAD PARAMETERS
F00F                    :LDTY     DS      1               ; LOAD TYPE (1 FOR BINARY, 0 FOR TEXT)
000C                    :FTDSP    EQU     12              ; TYPE DISPLACEMENT
000D                    :FADSP    EQU     13              ; MEM ADR DISPLACEMENT
000A                    :LBDSP    EQU     10              ; NO BLOCKS DISPLACEMENT
0008                    :LDSP     EQU     8               ; DISK ADR DISPLACEMENT
                        :*  DISK SAVE BUFFERS
F010                    :SRANG    DS      2               ; RANGE FOR SAVE
F012                    :SSTRT    DS      2               ; START ADR FOR SAVE
                        :*  FILE AREA PARAMETERS
F014                    :LCTRL    DS      1
F015                    :LPCNT    DS      1
F016                    :CLINE    DS      0               ; CURRENT LINE POINTER = LINC
F016                    :LINC     DS      2
2022                    :UCOMM    EQU     2022H           ; FDOS UTILITY ROUTINE
2028                    :FDOS     EQU     2028H           ; FDOS ENTRY POINT
                        :
                        :******** FILE DIRECTORIES AND SYSTEM TABLES ********
                        :
0008                    :NMLEN    EQU     8               ; NAME LENGTH
                        :*  LOCAL BINARY FILE DIRECTORY
000A                    :MAXBFIL  EQU     10              ; MAX NO OF BINARY FILES
F018                    :BFDIR    DS      MAXBFIL*NMLEN+4*MAXBFIL

PROGRAM ERRORS --       0
```

ARIAN SYSTEM RAM BUFFERS

```
                        ;*  LOCAL TEXT FILE DIRECTORY
000A                    ;MAXFIL  EQU     10              ; MAX NO OF FILES
0010                    ;FELEN   EQU     NMLEN+8         ; DIRECTORY ENTRY LENGTH
F090                    ;FILE0   DS      NMLEN
F098                    ;FOFP    DS      2
F09A                    ;EOFP    DS      2
F09C                    ;MAXL    DS      4
F0A0                    ;        DS      MAXFIL*FELEN-FELEN ; OTHER FILE ENTRIES
                        ;*  CUSTOMIZE COMMAND TABLE AND PARAMETERS
0014                    ;NCUST   EQU     20              ; 20 COMMANDS
F130                    ;CUSTT   DS      NCUST*6+1       ; 6 BYTES/CMND & NUMBER BYTE
                        ;* DEFINE BREAKPOINT REGION
0008                    ;NBR     EQU     8               ; NUMBER OF BREAKPOINTS
F1A9                    ;BRT     DS      3*NBR           ; BREAKPOINT TABLE
                        ;*  REGISTER SAVE REGION
F1C1                    ;REGSAV  DS      18              ; REGISTER SAVE AREA
F1D3                    ;HLSAV   DS      2               ; HL
F1D5                    ;SPSAV   DS      2               ; SP
F1D7                    ;PCSAV   DS      2               ; PC
                        ;
                        ;******** END OF FILE DIRECTORIES AND SYSTEM TABLES ********
                        ;
                        ;*  COMMAND LEVEL 3 (CL3) BUFFERS
F1D9                    ;SDRIV   DS      1               ; SAVE BUFFER FOR CURRENT DRIVE NUMBER
F1DA                    ;CDRIV   DS      1               ; COMMAND DRIVE NUMBER BUFFER
F1DB                    ;SBUF    DS      8               ; SAVE BUFFER FOR ORIGINAL FBUF
                        ;
                        ;*  EDITOR BUFFERS
F1E3                    ;INSP    DS      2               ; INSERT LINE POSITION
F1E3                    ;DELP    EQU     INSP            ; DELETE LINE POSITION
                        ;
F1E5                    ;HCON    DS      2
F1E5                    ;ADDS    EQU     HCON            ; FIND ADDRESS
F1E7                    ;ADDS1   DS      2               ; ADDR STORAGE SPACE
                        ;
                        ;*  FILE SYSTEM BUFFERS
F1E9                    ;FBUF    DS      NMLEN+1         ; FILE NAME BUFFER
F1F2                    ;FREAD   DS      2               ; FREE ADDRESS IN DIRECTORY
F1F4                    ;FEF     DS      1               ; FREE ENTRY FOUND FLAG
F1F4                    ;FOCNT   EQU     FEF             ; OUTPUT COUNTER
                        ;
                        ;*  COMMAND BUFFER
F1F5                    ;CBUF    DS      8               ; COMMAND AND SPECIAL CHARS
F1F9                    ;SPCHR1  EQU     CBUF+4
F1FA                    ;SPCHR2  EQU     CBUF+5
F1F9                    ;SPCHAR  EQU     SPCHR1          ; SPECIAL CHAR IMMEDIATELY AFTER CMND
                        ;*  COMMAND ARGUMENT BUFFERS
F1FD                    ;ABUF    DS      16              ; ASCII BUFFER
F20D                    ;BBUF    DS      6               ; BINARY BUFFER
```

PROGRAM ERRORS -- 0

ARIAN SYSTEM RAM BUFFERS

```
F213                    :S1BUF   DS      40              ; 1ST STRING BUFFER
F23B                    :S2BUF   DS      40              ; 2ND STRING BUFFER
                        :
F263                    :SCNT    DS      1
F264                    :LSAV    DS      1               ; TEMP BUFFER FOR LIST OPTION
F265                    :NL      DS      1               ; NL=NUMBER OF LINES TO PRINT
                        :
                        :*  ASSEMBLER BUFFERS
F266                    :LABA    DS      2               ; SYMBOL TABLE END ADDRESS
F268                    :ASPC    DS      2               ; ASSEMBLER PROGRAM COUNTER
                        :
F26A                    :ASMST   DS      2               ; ASSEMBLY START ADDRESS
F26C                    :ASMEN   DS      2               ; ASSEMBLY END ADDRESS
                        :
F26E                    :SYSERR  DS      1               ; ASSEMBLER ERROR FLAG
F26F                    :ASMRET  DS      2               ; ASSEMBLER RETURN ADDRESS
                        :
F271                    :PASI    DS      1               ; PASS INDICATOR
F271                    :PKEY    EQU     PASI            ; PASS KEY FOR MEMORY MANAGEMENT
F272                    :SCHR    DS      1               ; LENGTH OF STRING FOR COMPARE
F273                    :PNTR    DS      2               ; LINE POINTER STORAGE
                        :
F275                    :NOLA    DS      2               ; NUMBER OF LABELS
0180                    :MXLAB   EQU     384             ; MAXIMUM NUMBER OF LABELS
                        :
F277                    :OPER    DS      1               ; OPERAND STORAGE FOR SCAN
F278                    :OPRD    DS      2               ; OPERAND STORAGE
F27A                    :OPRI    DS      1               ; OPERAND FOUND INDICATOR
F27B                    :TEMP    DS      1
                        :
F1E3                    :APNT    EQU     INSP            ; ASSEMBLE LINE POINTER
F263                    :AERR    EQU     SCNT            ; ASSEMBLER ERROR PRINT SWITCH
F27C                    :OIND    DS      2               ; OUTPUT ADDRESS
                        :
                        :
0006                    :LLAB    EQU     6               ; LENGTH OF LABLES
                        :
                        :*  ADDITIONAL SPACE FOR FUTURE BUFFER EXPANSIONS
F27E                    :RESERVE EQU     $
F27E                    :        DS      0F300H-RESERVE-30
                        :
                        :*  STACK BUFFER
F2E2                    :        DS      30              ; STACK AREA
F300                    :STACK   DS      0               ; SET STACK POINTER
                        :
FE01                    :IBUF    EQU     0FE01H          ; INPUT BUFFER ADDRESS
0078                    :BUFLEN  EQU     120             ; NO OF CHARS PERMITTED IN ARIAN INPUT BUFFER
FDEC                    :OBUF    EQU     IBUF-21         ; OUTPUT BUFFER AREA
                        :
```

PROGRAM ERRORS -- 0

```
                        :*  EXTENDED BUFFER REGION - 0F300H AND BEYOND
                        :
F300                    :SYMT    EQU     $              ; START OF SYMBOL TABLE
F300                    :DIRT    EQU     SYMT           ; 1K DIRECTORY TABLE
F300                    :LOLD    EQU     SYMT           ; OLD LINE BUFFER FOR LINE EDITOR STORAGE
```

PROGRAM ERRORS -- 0

```
                        :********                                        ********
                        :******** ARIES-1 UTILITY PROM ROUTINE ADDRESSES ********
                        :********                                        ********
                        :*
                        :*  ARIES-1 UTILITY PACKAGE ROUTINE ADDRESSES
                        :*
D000                    :UTILITY EQU     0D000H          ; ORIGIN OF UTILITY
D111                    :ADR     EQU     UTILITY+111H    ; HL=HL+A ADDRESS ROUTINE
D108                    :MUL     EQU     UTILITY+108H    ; HL=HL*DE MULTIPLY ROUTINE
D108                    :DIV     EQU     UTILITY+108H    ; HL=HL/DE DIVIDE ROUTINE
D096                    :AHS1    EQU     UTILITY+096H    ; CATH
D093                    :BIN1    EQU     UTILITY+093H    ; CHTA
D03F                    :CRLF    EQU     UTILITY+03FH
D105                    :CAPS    EQU     UTILITY+105H
D099                    :EN      EQU     UTILITY+099H
D006                    :INIPI   EQU     UTILITY+006H    ; INIP
D07B                    :INK1    EQU     UTILITY+07BH
D021                    :INB     EQU     UTILITY+021H    ; INPB
D01B                    :INPUT   EQU     UTILITY+01BH
D01E                    :OUTPUT  EQU     UTILITY+01EH
D048                    :BSLSH   EQU     UTILITY+048H
D05D                    :PCHAR   EQU     UTILITY+05DH
D057                    :REM     EQU     UTILITY+057H
D07E                    :LMOV    EQU     UTILITY+07EH
D081                    :LMOVC   EQU     UTILITY+081H
D084                    :LMOVL   EQU     UTILITY+084H
D08D                    :PMOVL   EQU     UTILITY+08DH
D024                    :OUTB    EQU     UTILITY+024H    ; OUTB
D036                    :OUT3A   EQU     UTILITY+036H    ; OUT3
D060                    :PHOUT   EQU     UTILITY+060H    ; PA2HC
D069                    :DOUT    EQU     UTILITY+069H    ; PADC
D075                    :PDE     EQU     UTILITY+075H
D06C                    :PHL     EQU     UTILITY+06CH
D06F                    :PHLB    EQU     UTILITY+06FH
D078                    :PQ1     EQU     UTILITY+078H    ; PQ
D04E                    :PPRINT  EQU     UTILITY+04EH
D04B                    :PRINT   EQU     UTILITY+04BH
D042                    :BLK1    EQU     UTILITY+042H    ; PRBL
D09F                    :RANGE   EQU     UTILITY+09FH
D0AE                    :READ1   EQU     UTILITY+0AEH    ; READ
D087                    :RMOV    EQU     UTILITY+087H
D051                    :SCRNA   EQU     UTILITY+051H    ; SCRN
D0A5                    :TAPE    EQU     UTILITY+0A5H
D0A8                    :TABR    EQU     UTILITY+0A8H
D0A2                    :TABS    EQU     UTILITY+0A2H
D0BA                    :UREL    EQU     UTILITY+0BAH
D0B7                    :VREL    EQU     UTILITY+0B7H
D00F                    :GETCI   EQU     UTILITY+0FH
D015                    :GETCO   EQU     UTILITY+15H

PROGRAM ERRORS --       0
```

```
D012                    :RESCI   EQU     UTILITY+12H
D018                    :RESCO   EQU     UTILITY+18H
                        :*
                        :*  MCS ENTRY POINTS FOR UTIL
                        :*
D800                    :MCS     EQU     0D800H          ; ORIGIN OF MCS
D803                    :GETIN   EQU     MCS+03H
D818                    :UEXAM   EQU     MCS+18H
D81B                    :UDEPOS  EQU     MCS+1BH
D81E                    :UFIND   EQU     MCS+1EH
D824                    :USPTR   EQU     MCS+24H
D815                    :UCOPY   EQU     MCS+15H

PROGRAM ERRORS --       0
```

```
                        ASSEMBLY CONTROL AREA

                        :*
                        :*  ASMZ80 CONTROL AREA
                        :*
                        :       TITLE   'THE ARIAN OPERATING AND SOFTWARE DEVELOPMENT SYSTEM'
                        :       XREF
                        :*
                        :*  DISPLAY MESSAGE TO USER
                        :*
                        :       MESSAGE 'THE LAST BYTE OF ARIAN II IS LOCATED AT ADDRESS --'
  1FFE                  :       DISPLAY ENDALL
                        :       END
NO PROGRAM ERRORS

  PROGRAM ERRORS --     0
```

SYMBOL	VALUE	ADDR TYPE	REFERENCES
AADD	198E	ABSOLUTE	1960
AAND	19AD	ABSOLUTE	196C
ABUF	F1FD	ABSOLUTE	015C 01E0 01E7 01F1 01F8 01FF 0209 0210 0217 0221 02AF 028E 02D8 03DF 09E2 0BB1 0C0D 0D6E 0D7B 0D87 0E03 1035 1042 1066 112E 1193 119A 147E 176E 178C 17E2 1802 1806 182B 1847 1865 19D3
ACO1	14E6	ABSOLUTE	15A3
ADDS1	F1E7	ABSOLUTE	04A8 04C2 0A94 0A9A 0AAA 0ABE 0ACD 0ADC 0F1B 0F22
ADDS	F1E5	ABSOLUTE	0107 012D 02DB 02E9 04DC 0F8E 119D 1771 1779 178F 184A
ADEC1	0259	ABSOLUTE	026D
ADEC	0256	ABSOLUTE	19E0
ADR	D111	ABSOLUTE	02EF 02F8 0474 05DA 0857 087A 089C 08A3 08EE 09F3 0A14 0A2A 0A55 0A6D 0A7D 0AA5 0AAF 0AC3 0AE1 0B48 0B94 0C80 0E81 0E97 0F04 0F13 0F35 0F4E 0FD3 0FE4 117C
AERR	F263	ABSOLUTE	1388 1429
AHEX1	0272	ABSOLUTE	0283
AHEX	026F	ABSOLUTE	01EA 0202 021A 19E5
AHIGH	1988	ABSOLUTE	1978
AHS1	D096	ABSOLUTE	0279 1DE3
ALAB	1985	ABSOLUTE	194C
ALOW	1989	ABSOLUTE	1974
ALPS	1888	ABSOLUTE	0229 14A2 1844 19CE
ALP1	1890	ABSOLUTE	18A4 18AB
AMULT	1983	ABSOLUTE	197C
AOR	199D	ABSOLUTE	1964
AOUT1	141E	ABSOLUTE	1417
AOUT	1412	ABSOLUTE	140D 1A2C
AOU1	1435	ABSOLUTE	1423 1427 142D 1A08
AOU2	143D	ABSOLUTE	1442
APNT	F1E3	ABSOLUTE	13CB 13D1 13F3
ARIAN	0075	ABSOLUTE	0000
ASBL	18AE	ABSOLUTE	15A6 15D9 1618 162A 1651 1662 1680 16CB 16D8 170B
ASCN	18B1	ABSOLUTE	14F2 1506 1519 16A3
ASC0	18F5	ABSOLUTE	18D7 18DB 18DF 18E3 18E7 18EB 18EF
ASC1	18F8	ABSOLUTE	18D3
ASC2	1907	ABSOLUTE	18F3
ASC3	191D	ABSOLUTE	1912
ASC4	1926	ABSOLUTE	1943
ASC5	1945	ABSOLUTE	191F
ASC6	198F	ABSOLUTE	1981 1986 198C 199B 19A3 19AB 19B3
ASMEN	F26C	ABSOLUTE	08C4 09BB 12B5 12D6 1312 1734 174C
ASMRET	F26F	ABSOLUTE	1353 176A
ASMST	F26A	ABSOLUTE	0849 0870 0880 09B5 12AF 12CE 130C 13C5 163D 1745
ASM1	137D	ABSOLUTE	1375
ASM2	138C	ABSOLUTE	173F
ASM3	13D1	ABSOLUTE	1402 1410
ASM4	1404	ABSOLUTE	13FA

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
ASPC	F268	ABSOLUTE	13C2 148B 1490 14FB 1542 1549 1636 163A 1730 1816 181A 1918
ASSM0	135F	ABSOLUTE	1358
ASSM1	1372	ABSOLUTE	1369
ASSM	1352	ABSOLUTE	03E2 1CBB
ASTO	1717	ABSOLUTE	15C9 15CD 164E 16BC 16C0 16C8 1708 1710 1A1C
ASUB	1995	ABSOLUTE	1970
AUTO1	0BFE	ABSOLUTE	0C02
AUTO2	0C07	ABSOLUTE	0BF3 0BFB
AUTO	0BF0	ABSOLUTE	1CD9
AVAL1	1955	ABSOLUTE	1938
AVAL	1954	ABSOLUTE	191B 19B8
AXOR	19A5	ABSOLUTE	1968
BBUF	F20D	ABSOLUTE	01EE 0206 021E 03E8 03F2 0418 0447 0489 049E 0530 056F 05FC 0642 0655 0846 08FB 0988 09BE 09C6 09CD 09E8 09EB 09FA 0D77 0D8D 0E09 113F 1143 12AC 12B2 136F 1377 137A 137D 138F 15DE 15E2 160C 1611 1646 164A 1717 171C 1A58 1A78 1A7E 1A88 1B41 1DD1
BDEL	12EA	ABSOLUTE	0CF5
BDIRC	133E	ABSOLUTE	1350
BDIRL1	132A	ABSOLUTE	1330
BDIRL	1320	ABSOLUTE	1345
BDIRN	1348	ABSOLUTE	1325
BDIR	1309	ABSOLUTE	0C4E
BFDENT1	12C6	ABSOLUTE	12C1
BFDENT	12B8	ABSOLUTE	08CA 12A6 1760
BFDIR	F018	ABSOLUTE	126C 1291 12E3 1318
BFSCAN	126A	ABSOLUTE	0404 1288 12EA 12FA
BFSF	1285	ABSOLUTE	1276 1287
BFS1	126F	ABSOLUTE	128D
BFS2	1274	ABSOLUTE	127B
BFS3	1281	ABSOLUTE	129A
BFS4	1297	ABSOLUTE	129D
BINAD	161B	ABSOLUTE	1633
BINH	0285	ABSOLUTE	1545 154C 1620 1625 1723
BIN1	D093	ABSOLUTE	028A 0290 1DDD
BLK1	D042	ABSOLUTE	0356 0373 06C7 0C91 0CAF 0CB8 0CE1 0CE8 0E4C 0E62 123F 1242 1332 1335 1B22 1B2F 1B32 1DA7
BL1	1AE5	ABSOLUTE	1AF0
BL2	1AEE	ABSOLUTE	1AE8
BL3	1AF5	ABSOLUTE	1AEC
BNAME	12FA	ABSOLUTE	0C22
BOFP	F098	ABSOLUTE	02DE 088F 0917 098B 0BD6 0DC2 0DE3 0DF2 0E0C 1152 11B1 13C8 1DC5
BREAK	1A32	ABSOLUTE	1CC7
BREKE	1A8F	ABSOLUTE	1A34
BRKEL	1A97	ABSOLUTE	1AA3
BRKEZ	1AA0	ABSOLUTE	1A9A
BRKPD	1A75	ABSOLUTE	1A39

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
BRKPE	1AE0	ABSOLUTE	1A85
BRKP	1ABD	ABSOLUTE	0008
BRT	F1A9	ABSOLUTE	008F 1A46 1A8F 1AA6 1AE0
BSCR	12E3	ABSOLUTE	007D 0549
BSLSH	D048	ABSOLUTE	0766 077E
BUFLEN	0078	ABSOLUTE	0716
B1	1A49	ABSOLUTE	1A52
B2	1A57	ABSOLUTE	1A4D
B3	1A6B	ABSOLUTE	1A5E 1A63
CAPS	D105	ABSOLUTE	0184 01B1 04B9 06EC 0776 0C42 188D 189C
CBUF	F1F5	ABSOLUTE	017B F1FD
CDCONT	058F	ABSOLUTE	0587
CDERR1	0589	ABSOLUTE	04D2 04E2 05C9
CDERR	057D	ABSOLUTE	0573 0577
CDRIV	F1DA	ABSOLUTE	00C3 0558 0564 056B 0579 0583 058A 05EE
CDSET	056F	ABSOLUTE	0562
CEXADR	F00D	ABSOLUTE	05B0 05E9
CHLDE	035F	ABSOLUTE	0DD0 0F57 0F61
CKA11	02AF	ABSOLUTE	0040 02A5 03B9 03D2 043F 055F 09B0 0BF0 1029 12A3 1366 1B3C
CKA1	02A5	ABSOLUTE	05F9 06B8 0B51 1187 1A41 1A75
CKA21	02BE	ABSOLUTE	0043 02B4 102E 1372
CKA2	02B4	ABSOLUTE	09C3 12A9
CKFN1	029F	ABSOLUTE	0046 0295 03B4 03FF 090D 11FB 175B
CKFN2	02A2	ABSOLUTE	02B2 02C1
CKFN	0295	ABSOLUTE	042E 0843 08E3 0958 096B 097E 0C1C 0CEF 12A0
CLBL	1AAB	ABSOLUTE	1ABA
CLERA	12DE	ABSOLUTE	0491 0C2E 12E8 12F8
CLER1	13EC	ABSOLUTE	13E5
CLER	13E4	ABSOLUTE	13EA
CLINE	F016	ABSOLUTE	0669 0675
CLOAD	05B4	ABSOLUTE	05AD
CLRA	00FB	ABSOLUTE	00FE 12E0
CLRB	1AA6	ABSOLUTE	1A3E
CLRZ	00FA	ABSOLUTE	0094 0172 0551
CL2	1ABA	ABSOLUTE	1AB6
CMNDS	055A	ABSOLUTE	0075 0568
CMND1	0583	ABSOLUTE	0126
CMND	055F	ABSOLUTE	1D3F
COMM0	011E	ABSOLUTE	0116
COMM1	0129	ABSOLUTE	011C
COMM	010B	ABSOLUTE	00EF
COMS	012D	ABSOLUTE	0119 0123 013E 04DF
COM01	031A	ABSOLUTE	0322
COM02	031B	ABSOLUTE	032B
COM03	031F	ABSOLUTE	031C
COM0	0315	ABSOLUTE	02F2 0F94 11A4

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
COM1	0326	ABSOLUTE	106A
COND	176E	ABSOLUTE	17D7
CONT	1B39	ABSOLUTE	1CCD
COPC	1779	ABSOLUTE	1776 1787 1797 179E 17A4 17AE 17B6 17BF 17C7 17D2 1853
COP1	1789	ABSOLUTE	177E
CREL	D0B7	ABSOLUTE	1DF5
CRLFP	10A7	ABSOLUTE	1074 1438
CRLF	D03F	ABSOLUTE	00C0 036C 038A 03F5 041D 0515 0554 060B 06BE 06DA 0729 0B69 0C9F 0E38 1214 1263 1309 1327 1742 1A94 1B10 1B39 1D8F
CSTL1	052C	ABSOLUTE	0464
CSTL2	046B	ABSOLUTE	045A
CSTN1	04A3	ABSOLUTE	04A5
CSTN2	04B4	ABSOLUTE	04C0
CSTN3	04C2	ABSOLUTE	04CA
CSTN4	04C7	ABSOLUTE	04B7
CTAB	1C99	ABSOLUTE	011E
CTS	04CC	ABSOLUTE	0496 04E6
CUSDL	04F2	ABSOLUTE	04F4
CUSE1	0526	ABSOLUTE	0452 0D84 12BB 1479 1A54
CUSML	04F8	ABSOLUTE	0504
CUSM1	04FD	ABSOLUTE	0502
CUSP1	0513	ABSOLUTE	0523
CUSTD	04E6	ABSOLUTE	043C
CUSTL	050B	ABSOLUTE	0425
CUSTN	0496	ABSOLUTE	0437
CUSTP	0518	ABSOLUTE	051D
CUSTS	0506	ABSOLUTE	0097 042A
CUSTT	F130	ABSOLUTE	010B 044A 04CC 04E9 0507 050B
CUST0	044A	ABSOLUTE	0442
CUST1	045C	ABSOLUTE	0469
CUST2	046F	ABSOLUTE	0477
CUST3	0479	ABSOLUTE	0470 04C5
CUST4	047E	ABSOLUTE	0487
CUST5	0489	ABSOLUTE	0494
CUST6	0491	ABSOLUTE	0481
CUST	0423	ABSOLUTE	1CC1
DAT1	1521	ABSOLUTE	14D0
DAT2A	15EC	ABSOLUTE	1521
DAT2	15E9	ABSOLUTE	1587
DCOMM	2022	ABSOLUTE	02C3
DCOM	02C3	ABSOLUTE	0069 05F6 08B3 0AF6 0AFF 0B19 0B22 0C66
DELDF	0958	ABSOLUTE	1D2D
DELL	1187	ABSOLUTE	1CB5
DELP	F1E3	ABSOLUTE	1190 11A7 11EA
DEL1	119D	ABSOLUTE	1198
DEL2	11B8	ABSOLUTE	11C3

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR	TYPE	REFERENCES
DEL3	11D2	ABSOLUTE		11CF 11DD
DEL4	11DB	ABSOLUTE		11BE
DEL5	11DC	ABSOLUTE		11CA
DEOL	0785	ABSOLUTE		076C
DIRD	0C69	ABSOLUTE		1CEB
DIRPL	0C79	ABSOLUTE		0C8B 0CDB
DIRP2A	0C91	ABSOLUTE		0C94
DIRP2	0C8D	ABSOLUTE		0C7C
DIRP3	0CA5	ABSOLUTE		0C9A 0CAB
DIRP4	0CD1	ABSOLUTE		0CCA 0CE6
DIRP5	0CDD	ABSOLUTE		0CC3
DIRP6	0CE1	ABSOLUTE		0CE4
DIRT	F300	ABSOLUTE		0A4A 0A5B 0B0B 0B28 0C60 0C6C
DIR	0C56	ABSOLUTE		0A04 0B25 0C69
DIV	D10B	ABSOLUTE		197E
DLDF2	0963	ABSOLUTE		0966
DNAME	096B	ABSOLUTE		1CE5
DOUT	D069	ABSOLUTE		0CB3 1DA1
DRIVEC1	08D0	ABSOLUTE		09AD
DRIVEC	08CD	ABSOLUTE		0C56
DRIVEN	F00A	ABSOLUTE		007A 00CB 05B4 05BD 05C6 08D8 08DD
DRIVE	08DC	ABSOLUTE		08AE 0AEE 0B12 0C5D
DRSDF	0868	ABSOLUTE		0407 040D 0852 08E9 091F 095E 0971 0C28 0CFB 12ED 12F3 12FD 1306
DSAVE	097A	ABSOLUTE		0058 1D27
DSCAN	0B25	ABSOLUTE		0049 05C0 084F 08E6 095B 096E 09EE 0A07
DSCNF	0B4D	ABSOLUTE		0B44
DSCNL1	0B3F	ABSOLUTE		0B3B
DSCNL	0B33	ABSOLUTE		0B3D
DSCNN	0B42	ABSOLUTE		0B37
DSCN1	0B2D	ABSOLUTE		0B4B
ENDALL	1FFE	ABSOLUTE		F300
EN	D099	ABSOLUTE		1DE9
EOFP	F09A	ABSOLUTE		0912 0992 0B5B 0BDC 0C15 0DBE 0DF7 0E0F 0FA6 0FB0 0FCE 0FD8 0FED 11AC 11F2 1DCB
EOF1	02FE	ABSOLUTE		02E6 068E
EOF	02FD	ABSOLUTE		1056
EOLT	068E	ABSOLUTE		0673 0684
EORP	00EC	ABSOLUTE		00E5
EOR	00BA	ABSOLUTE		0066 00DD 00E1 00EA 00F2 00F7 02E3 0302 0341 0369 03EB 0537 068B 0DE0 1767 1B0D
EPASS1	1742	ABSOLUTE		172E
EPASS2	1763	ABSOLUTE		175E
EPASS	172A	ABSOLUTE		13DA 14E3 159D
EQUL	0FBC	ABSOLUTE		0FB9
EQUS	1511	ABSOLUTE		1504
EQU1	1506	ABSOLUTE		14CC

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
EQU2	1618	ABSOLUTE	1582
ERRA	1A0B	ABSOLUTE	192D 193E 1947 1951 198A
ERRD	1A27	ABSOLUTE	1467
ERRL	1A23	ABSOLUTE	156B 1571
ERRM	1A03	ABSOLUTE	150E
ERRO1	1A11	ABSOLUTE	1A25
ERRO2	1A1B	ABSOLUTE	1A20
ERRO	1A0F	ABSOLUTE	1824
ERRR1	19EA	ABSOLUTE	1A01
ERRR	19E8	ABSOLUTE	1654 165D 1665 166A 1670 167B 1683 1689 16A6 16AC 16DB 16E1 1702
ERRS1	19F3	ABSOLUTE	19FD 1A0D
ERRS	19F1	ABSOLUTE	15FE 16F4 18FD 190C 19C2
ERRU	19FB	ABSOLUTE	198C
ERRV	19FF	ABSOLUTE	16D1
EXADR	F00B	ABSOLUTE	009D 03E5 0444 0884 1380 15AF
EXECA	03D2	ABSOLUTE	03B7
EXECB	03FF	ABSOLUTE	03B2
EXECD	03E5	ABSOLUTE	03CB 0402
EXECR	03EB	ABSOLUTE	03D5 041B
EXEC1	03D9	ABSOLUTE	03C1
EXEC2	03DC	ABSOLUTE	03D0 03D7
EXEC	03B0	ABSOLUTE	1CA3
EXIT	0554	ABSOLUTE	1D1B
EXT	1C91	ABSOLUTE	05A5
EX2	15A6	ABSOLUTE	1593
FADSP	000D	ABSOLUTE	0878
FBUF	F1E9	ABSOLUTE	016D 01A1 029F 0431 045C 047B 04AF 058F 059D 05CF 067F 0AD1 0B2D 0D61 0D9E 0EB8 1247 1271 12C6
FCHKE	0952	ABSOLUTE	0933 0937 094D
FCHK0	091C	ABSOLUTE	0910
FCHK1	092E	ABSOLUTE	091A
FCHK	090A	ABSOLUTE	1D33
FCK1	0949	ABSOLUTE	0946
FCK	093F	ABSOLUTE	092E 0950 0BD9
FCML1	0F66	ABSOLUTE	0F5A
FCPIL	0F4C	ABSOLUTE	0EFF
FCPI1	0F01	ABSOLUTE	0F5C 0F64
FCPI2	0F02	ABSOLUTE	0F73
FCPI	0EFC	ABSOLUTE	0F07
FCURE	F008	ABSOLUTE	0F0E 0F30 0F6B
FCUR	F006	ABSOLUTE	0EF0 0F5E 0F67
FDELL	0D14	ABSOLUTE	0D18
FDELP	0D25	ABSOLUTE	0D03
FDEL	0CEE	ABSOLUTE	08C7 1D03
FDLF1	0D44	ABSOLUTE	0D48
FDLPF	0D3E	ABSOLUTE	0D2F

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
FDLP1	0D2D	ABSOLUTE	0D32
FDLP2	0D38	ABSOLUTE	0D38 0D4D
FDL1	0D20	ABSOLUTE	0D22
FDOS	2028	ABSOLUTE	0557
FECHK	02CD	ABSOLUTE	02D5 090A 097B 0C07 0F76 1125 1363
FEET	0E57	ABSOLUTE	0E33
FEF	F1F4	ABSOLUTE	0D7E 0EB0 0ED4 F1F5
FELEN	0010	ABSOLUTE	054F 0D06 0D0E 0D1A 0D1D 0D25 0D28 0DAC 0E57 0ED9 F0A0
FENTF	0D87	ABSOLUTE	0D6C
FFIXN1	0BE9	ABSOLUTE	0BED
FFIXN	0BDF	ABSOLUTE	0BE1
FFIX	0BD6	ABSOLUTE	0608 088D 112B
FILEB	12A0	ABSOLUTE	0D5E
FILE0	F090	ABSOLUTE	02CD 054C 0CFE 0D06 0D1A 0D28 0D34 0D41 0DA9 0E1A 0E72 0EB5 0EF3
FILE1	0D7E	ABSOLUTE	0D72
FILE	0D5C	ABSOLUTE	004C 03D9 0602 084C 1CA9
FINDL	0669	ABSOLUTE	0692
FINDS	0655	ABSOLUTE	1D51
FIND1	02DE	ABSOLUTE	0FB4
FIND2	02E6	ABSOLUTE	02FB 11E3
FIND3	02F7	ABSOLUTE	11E6
FIND	02D5	ABSOLUTE	004F 0658 06BB 0B54 0BF5 104F 118A
FOCNT	F1F4	ABSOLUTE	0E1E 0E5B
FOUL	0E1A	ABSOLUTE	0C53
FOUTL	0E1E	ABSOLUTE	0E5F
FPRI1	0E3B	ABSOLUTE	0E41
FPRI2	0E4F	ABSOLUTE	0E54
FPRI	0E35	ABSOLUTE	0E28 0E2D
FPRMP	0E18	ABSOLUTE	0D65 0E07
FPRM1	0DAE	ABSOLUTE	0DB6
FPRM	0DA9	ABSOLUTE	0D8B 0D92
FPTRP	0E62	ABSOLUTE	051F 0E43 0E46
FREAD	F1F2	ABSOLUTE	0D9A 0ED0
FSEA1	0E88	ABSOLUTE	0EE0
FSEA2	0ED9	ABSOLUTE	0EE4
FSEA3	0EE2	ABSOLUTE	0EC4 0EC9
FSEA	0EAF	ABSOLUTE	005E 03BE 091C 0C25 0CF8 0D68
FSP	0D9A	ABSOLUTE	0D82
FTDSP	000C	ABSOLUTE	0855 08EC 09F1
GETIN	D803	ABSOLUTE	0616
GETSP	0390	ABSOLUTE	1077 1096 141E
GNO1	1161	ABSOLUTE	1163
GNO	115E	ABSOLUTE	116A
GT	0FE0	ABSOLUTE	0FCC
HCON	F1E5	ABSOLUTE	F1E7
HLSAV	F1D3	ABSOLUTE	1ABD 1AC9 1B5E

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
HOUT	D060	ABSOLUTE	1B26 1B2B 1D9B
IBPMS	1A65	ABSOLUTE	1AF2
IBUF	FE01	ABSOLUTE	0104 0175 059A 06D4 0716 07E4 0805 080A 0828 082E 0835 0B73 0B8D 0BA5 0F7B 0FA3 0FC3 0FF6 0FFF 13E2 1450 14FE 1509 1556 1DD7 F300
INIPI	D006	ABSOLUTE	00B1
INITA	009A	ABSOLUTE	0004 009A 1D4B
INK1	D07B	ABSOLUTE	00F4
INK	00F4	ABSOLUTE	0072 0CA2 106F 13FF 1407
INPUT	D01B	ABSOLUTE	0686 1D83
INSE	0B57	ABSOLUTE	0C04 0C18
INSFX	0BCE	ABSOLUTE	0B79
INSLL	F008	ABSOLUTE	0B5E 0B90 0B97 F00A
INSL1	0B7D	ABSOLUTE	0B88
INSL2	0B8A	ABSOLUTE	0B80 0B84
INSL	0B69	ABSOLUTE	0BCA
INSNL	F006	ABSOLUTE	0B57 0BA2 0BC3 F00A
INSP	F1E3	ABSOLUTE	0FC0 0FF0 F1E5 F27C
INSR	0FB4	ABSOLUTE	0F97
INS	0B51	ABSOLUTE	1D39
IN8	D021	ABSOLUTE	06E1 06E9 073B 0773 0793 07B9 07EE
ISL3A	0BB6	ABSOLUTE	0BBA
JREL	D0BA	ABSOLUTE	1DEF
LBACE	0819	ABSOLUTE	07E7
LBACK1	083E	ABSOLUTE	0838
LBACK	0835	ABSOLUTE	06F3 07A3 07C9
LBAC	07E4	ABSOLUTE	07F3
LBDSP	000A	ABSOLUTE	089A 0A12 0A28 0AC1
LCHK	14B0	ABSOLUTE	146A 156E
LCHR1	0710	ABSOLUTE	07A8 07D7
LCHR	070B	ABSOLUTE	0705 0744 074F 075D
LCTRL	F014	ABSOLUTE	0C73 0C83 0C87 0CD3 0CD7 1026 104C 105C
LDBF0	0888	ABSOLUTE	0876
LDBF	086E	ABSOLUTE	0860
LDEL	0769	ABSOLUTE	077B
LDIR	0C4C	ABSOLUTE	1CF1
LDSP	0008	ABSOLUTE	08A1 0A7B 0AAD 0ADF
LDTF	088F	ABSOLUTE	085C
LDTY	F00F	ABSOLUTE	03C6 088A 0894 0889
LEDCE	0811	ABSOLUTE	0800
LEDC00	06F8	ABSOLUTE	06F1
LEDC0	06EC	ABSOLUTE	06E7 0782 07FB
LEDC1	0701	ABSOLUTE	06FA
LEDC2	072E	ABSOLUTE	0703
LEDC3	0747	ABSOLUTE	0752
LEDC4	0754	ABSOLUTE	0739
LEDC5	0758	ABSOLUTE	0760

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR	TYPE	REFERENCES
LEDC6	0762	ABSOLUTE		0756
LEDC7	0788	ABSOLUTE		0764
LEDC8	0785	ABSOLUTE		078D
LEDC9	07DC	ABSOLUTE		07B7
LEDC	06E9	ABSOLUTE		06F6 0708 0742 0749 074D 075B 0788 07B2 07BE 0816 081C
LEDDB	081F	ABSOLUTE		079C
LEDD	0823	ABSOLUTE		0734 07C3
LEDIT	06B8	ABSOLUTE		1D0F
LEDTO	068E	ABSOLUTE		080E
LEDX	0729	ABSOLUTE		0730
LEOLE	0722	ABSOLUTE		0719
LEOL	071C	ABSOLUTE		070E 0785 0819
LFIXL	1007	ABSOLUTE		100C
LFIXR	0F8E	ABSOLUTE		101A
LFIX	0FFF	ABSOLUTE		0F83 0F88 1014
LICK	0F7F	ABSOLUTE		0F8C
LINC	F016	ABSOLUTE		114F 116D
LINECNT	0396	ABSOLUTE		067C 0C9C 1059 1220 1447
LINE	0F76	ABSOLUTE		0052 00E7 0832
LINIT	03A9	ABSOLUTE		0666 0C76 1022 11F8 13CE
LINK1	D000	ABSOLUTE		0420
LINK	041D	ABSOLUTE		1D09
LINS0	07A8	ABSOLUTE		07A1 07D4
LINS1	07AE	ABSOLUTE		0798
LINS	0793	ABSOLUTE		07A6 07AC
LISTP	101D	ABSOLUTE		1D57
LISTO	1052	ABSOLUTE		1072
LIST	1022	ABSOLUTE		1CAF
LLAB	0006	ABSOLUTE		1227 1245 125E 13A1 147C 186D
LMAX	1C95	ABSOLUTE		0C0A
LMOVC	D081	ABSOLUTE		08C0 1047 12CB
LMOVL	D084	ABSOLUTE		0DA5 0DEB 0F2C
LMOV	D07E	ABSOLUTE		0FAB 0FE8 0FFB 11EF 13EE
LOADE	0862	ABSOLUTE		03CD
LOADF	0897	ABSOLUTE		088D
LOAD	0843	ABSOLUTE		0055 03C3 1D15
LODM	0305	ABSOLUTE		032D 0F9A 11D2
LOLD	F300	ABSOLUTE		06C1 06D7
LPCNT	F015	ABSOLUTE		039E 03A2 03AC 0A60 0A70 0A74 0E77 0E9A 0E9E 131D 133E 1342
LPRIN	07FE	ABSOLUTE		07DE
LREP0	07CE	ABSOLUTE		07C7
LREP1	07D6	ABSOLUTE		07D1
LREP	07B9	ABSOLUTE		07CC 07DA
LSAV	F264	ABSOLUTE		15B6 15BF
LSTALL	1035	ABSOLUTE		102C
LSTA1	103A	ABSOLUTE		103D

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
LSTRT	06D4	ABSOLUTE	072C
LST0	104A	ABSOLUTE	1031
LST1	104F	ABSOLUTE	1033
LST2	1583	ABSOLUTE	1588
LT	0FCE	ABSOLUTE	0FC7
MAXBFIL	000A	ABSOLUTE	126A 12E6 131B F018
MAXFIL	000A	ABSOLUTE	054F 0C51 0D2B 0E75 0EB3 0EF6 128F F0A0
MAXL	F09C	ABSOLUTE	0BE4 0E15 0F91 0F9D 103F 11A1 11D5 11DF
MCS	D800	ABSOLUTE	F300
MESS1	036C	ABSOLUTE	0366 0B05
MESS	0366	ABSOLUTE	003B 005B 029C 02AC 02BB 02CA 03FC 0529 0580 058C 0865 086B 093C 0955 09A7 0A3C 0D97 1184 1A68
MLAB	1481	ABSOLUTE	1485
MPNT	16EA	ABSOLUTE	16A0
MS0	1B62	ABSOLUTE	00B4
MS13	1B9E	ABSOLUTE	1AFD
MS1	1B71	ABSOLUTE	0299
MS22	1BC2	ABSOLUTE	0862
MS25	1BD0	ABSOLUTE	0939
MS26	1BDB	ABSOLUTE	0952
MS27	1BE6	ABSOLUTE	057D
MS2	1B77	ABSOLUTE	02A9
MS31	1BF3	ABSOLUTE	09A4
MS32	1BFB	ABSOLUTE	0346
MS33	1C03	ABSOLUTE	0B02
MS35	1C0E	ABSOLUTE	053A
MS36	1C19	ABSOLUTE	0868
MS38	1C28	ABSOLUTE	0526
MS39	1C37	ABSOLUTE	0D94
MS3	1B7E	ABSOLUTE	02B8
MS40	1C3E	ABSOLUTE	02C7
MS4	1B89	ABSOLUTE	03F9
MS50	1C47	ABSOLUTE	1181
MS51	1C51	ABSOLUTE	0A39
MS5	1B93	ABSOLUTE	0589
MS60	1C5B	ABSOLUTE	13B5
MS61	1C66	ABSOLUTE	1737
MS72	1C71	ABSOLUTE	0038
MS80	1C7B	ABSOLUTE	0723
MS92	1C81	ABSOLUTE	1A65
MS93	1C8A	ABSOLUTE	0DD5
MUL	D108	ABSOLUTE	1983
MXLAB	0160	ABSOLUTE	1473
NBR	0008	ABSOLUTE	0092 1A44 1A92 1AA9 1AE3 F1A9
NCHR	F272	ABSOLUTE	0110 0130 04D8
NCOM	0021	ABSOLUTE	0121

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
NCUST	0014	ABSOLUTE	044F F130
NLST2	158F	ABSOLUTE	158F
NL	F265	ABSOLUTE	00AE 03A9
NMLEN	0008	ABSOLUTE	01A4 0410 0922 0AD4 0B30 0C2C 0DA1 0E22 0E36 0E57 0E7F 0E95 0EBB 0ECC 0ED9 0EE6 0F02 0F11 0F33 0F4C 126F 127D 1294 12C9 12E6 12F6 1320 F018 F090 F1E9
NOLA	F275	ABSOLUTE	00A3 1201 1256 1394 1470 1494 1498 185D
NORM1	0333	ABSOLUTE	033C
NORM	032D	ABSOLUTE	0224 0231
NOVR	11E3	ABSOLUTE	11AA
NSYM	0019	ABSOLUTE	1206 120F 1D5F 1D60
NUMS	19CE	ABSOLUTE	194E
NUM1	19E0	ABSOLUTE	19DC
NUM2	19E3	ABSOLUTE	19D8
NXT1	18B8	ABSOLUTE	1992
NXT2	18C5	ABSOLUTE	1905
OBUF	FDEC	ABSOLUTE	13DF 140A 1412 142F 1435 14F5 153F 15A9 161C 162D 19ED 19F3 1A05 1A11 1A29
OCNT	17FF	ABSOLUTE	17AC 17CF
OCN0	1810	ABSOLUTE	1829
OCN1	1813	ABSOLUTE	14E9 14EF 15D6 15EF
OCN2	1816	ABSOLUTE	15E6
OERR	1824	ABSOLUTE	14AA 17DA 17F4 180D 1830
OIND	F27C	ABSOLUTE	1550 171F 1726
OPAD	17F9	ABSOLUTE	17F0
OPCD	178C	ABSOLUTE	14A7 14AD
OPC	149B	ABSOLUTE	1459 155F 1574 1A2F
OPER	F277	ABSOLUTE	18C2 18F5 195C
OPRD	F278	ABSOLUTE	18B4 1955 198F 19C5
OPRI	F27A	ABSOLUTE	18B8 18F8 1902 1908 1959 19BE
OP1	17A7	ABSOLUTE	17A1 17E9
OP2	17AA	ABSOLUTE	17B4 17BC 17C5
OP4	17FA	ABSOLUTE	17D5
OP5	17FD	ABSOLUTE	1464 146D
ORG1	14F2	ABSOLUTE	14C8
ORG2	162A	ABSOLUTE	157D
OTAB	1DF7	ABSOLUTE	1792
OUTAP3	111E	ABSOLUTE	1110 1115
OUTAP	110B	ABSOLUTE	1087 10A1 10A9 10AE 1083 10B6 10DB 10EA 10F3 1104 143E
OUTLB0	10FC	ABSOLUTE	10E3 10F1
OUTLB1	10FE	ABSOLUTE	10D0 10F6
OUTLB2	1104	ABSOLUTE	1107
OUTLNB	10D6	ABSOLUTE	108B 10C0 10CB 10FA
OUTLNC	10EF	ABSOLUTE	10D9
OUTLNE	108D	ABSOLUTE	1082
OUTLN1	1086	ABSOLUTE	108B
OUTLN2	109D	ABSOLUTE	1090 1094 10A5 10CE 10D4
OUTLN3	1089	ABSOLUTE	109B

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
OUTLN4	10D0	ABSOLUTE	10C8
OUTLN	1084	ABSOLUTE	107C 1444
OUTPUT	D01E	ABSOLUTE	00C8 00D0 0519 06CB 0710 0CBE 0E50 1119 122F 1B14 1B19 1B1E 1D89
OUTQ	10DB	ABSOLUTE	10E7
OUT3A	D036	ABSOLUTE	1120
OUT8	D024	ABSOLUTE	06FE 076F 07EB 0813 083B 0840 0CA6 0E3C 132B
PACK1	0EED	ABSOLUTE	0E00 0F4A
PARAME	03F9	ABSOLUTE	016A 03BC 0662
PARSE	016D	ABSOLUTE	006C 0166
PAR10	0229	ABSOLUTE	01E3 01FB 0213
PAR11	0231	ABSOLUTE	01F4 020C
PAR12	0239	ABSOLUTE	01D1 01DC 0247 024B
PAR13	0249	ABSOLUTE	0243
PAR14	024E	ABSOLUTE	023F
PAR1	017E	ABSOLUTE	018A
PAR2	018C	ABSOLUTE	0182
PAR3	018D	ABSOLUTE	0193
PAR4	01A7	ABSOLUTE	01B9
PAR5	01BB	ABSOLUTE	01AB 01AF
PAR6	01BD	ABSOLUTE	01C3
PAR7	01C5	ABSOLUTE	01B7 01BE
PAR8	01CA	ABSOLUTE	019F
PAR9	01E0	ABSOLUTE	01CC 01D7
PASI	F271	ABSOLUTE	13BC 13F6 144D 172A 1810 1833 1A14 F272
PASS1	144A	ABSOLUTE	13FC
PASS2	153F	ABSOLUTE	1404
PCHAR	D05D	ABSOLUTE	00D3 036F 060E 06DD 078F 07AE 07E0 07F6 081F 086C 1235
PCSAV	F1D7	ABSOLUTE	1A78 1A81 1A8B 1AC2 1ADC 1B02 1B44 1B5A
PDEADR1	0353	ABSOLUTE	035D
PDEADR	0350	ABSOLUTE	1239 1A9D
PDEP	0CE8	ABSOLUTE	0CCE
PDE	D075	ABSOLUTE	0353 0CEB
PEDADR	0359	ABSOLUTE	0E66 1338 133B
PEM	D057	ABSOLUTE	063C 071C
PFMS	09A4	ABSOLUTE	02D2
PHLB	D06F	ABSOLUTE	130F 1748 174F 1758 1DBF
PHL	D06C	ABSOLUTE	1315
PKEY	F271	ABSOLUTE	0EF9 0F09 0F70
PNN	053A	ABSOLUTE	04AB 0C33
PNTR	F273	ABSOLUTE	0195 0379 0385 1453 1480 148B 1527 1559 15F8 16EA 16F1 1898 188B 1915 1927 1935
PPRINT	D04E	ABSOLUTE	1DB3
PQ1	D078	ABSOLUTE	033E 0DDC
PQ	033E	ABSOLUTE	034D 03A6
PREPMS	0345	ABSOLUTE	052C 0A0C 12C3
PRINT	D04B	ABSOLUTE	1DB9

PROGRAM ERRORS -- 0

SYMBOL	VALUE ADDR	TYPE	REFERENCES
PRLINE	1B10	ABSOLUTE	1B05 1B0A
PRL1	1B13	ABSOLUTE	1B36
PSEU	182B	ABSOLUTE	179A
PSU1	14BF	ABSOLUTE	1837
PSU2	1577	ABSOLUTE	183A
PSZ81	14EC	ABSOLUTE	14C5
PSZ82	15C6	ABSOLUTE	157A
P1	1B47	ABSOLUTE	1B3F
RANGE	D09F	ABSOLUTE	0B62 0DC5 0F25 1753
RCVR	05F9	ABSOLUTE	1CFD
READ1	D0AE	ABSOLUTE	0101
READ	0101	ABSOLUTE	006F 00D7 0540 0B70
REGSAV	F1C1	ABSOLUTE	1B47
RENME	1138	ABSOLUTE	0BD3
RESCI	D012	ABSOLUTE	064F
RESCO	D018	ABSOLUTE	0652
RESERVE	F27E	ABSOLUTE	F27E
RES1	1519	ABSOLUTE	14E1
RES21	15E5	ABSOLUTE	151E 152D 1534 1539 1605 1609
RES2	15D9	ABSOLUTE	159B
RMOVL	D08D	ABSOLUTE	0B9F
RMOV	D087	ABSOLUTE	0FDB
RNERR	0A39	ABSOLUTE	09DD 118D
RNMEE	0C2B	ABSOLUTE	0974 1303
RNMES	0C3A	ABSOLUTE	0C36
RNME	0C1B	ABSOLUTE	1CDF
RNMS1	0C3B	ABSOLUTE	0C49
RNUME	1181	ABSOLUTE	1178
RNUML	1155	ABSOLUTE	117F
RNUM0	112E	ABSOLUTE	1128
RNUM1	113F	ABSOLUTE	1133 1136
RNUM2	114F	ABSOLUTE	113D 1147 114A
RNUM	1125	ABSOLUTE	0061 1CD3
RPDIR	0B08	ABSOLUTE	0902 0907 0968 0977 0A01
RTAB	1FEA	ABSOLUTE	1850
SASC1	1524	ABSOLUTE	14DD
SASC2	15F2	ABSOLUTE	1597
SASL1	1530	ABSOLUTE	153D
SASL2	1601	ABSOLUTE	1616
SAVD1	0A46	ABSOLUTE	0A48
SAVD	0A3F	ABSOLUTE	0A1C 0A22
SAVEB1	09C3	ABSOLUTE	09B3
SAVEB2	09C6	ABSOLUTE	09C1
SAVEB3	09EE	ABSOLUTE	09E6
SAVEB	09AA	ABSOLUTE	0989
SAVS	0ADF	ABSOLUTE	0A36

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
SAV1A	0A24	ABSOLUTE	0A1E
SAV1B	0A32	ABSOLUTE	0A2F
SAV1	0A04	ABSOLUTE	09A2 09DF
SAV2A	0A6D	ABSOLUTE	0A8F
SAV2D	0A5A	ABSOLUTE	0A52
SAV2L	0A4F	ABSOLUTE	0A58
SAV21	0A66	ABSOLUTE	0A77
SAV22	0A7B	ABSOLUTE	0A69
SAV23A	0ABE	ABSOLUTE	0ABB
SAV23	0A9A	ABSOLUTE	0A79
SAV24	0AD6	ABSOLUTE	0ADA
SAV2	0A4A	ABSOLUTE	0A0A
SBLK1	037C	ABSOLUTE	0251 0388
SBLK2	0384	ABSOLUTE	01C5 037F
SBLK	0379	ABSOLUTE	0198 0234 149E 14BF 15F2 18AE
SBUF	F1D8	ABSOLUTE	0592 05CC
SCANC1	124B	ABSOLUTE	1253
SCANC2	1252	ABSOLUTE	124E
SCANL1	1260	ABSOLUTE	1261
SCANL	1868	ABSOLUTE	125A
SCAN	1245	ABSOLUTE	11FE
SCNT	F263	ABSOLUTE	F27C
SCRNA	D051	ABSOLUTE	0376 038D 1DAD
SCRN0C	1074	ABSOLUTE	0678 1053
SCRN0	1077	ABSOLUTE	
SCRN	038A	ABSOLUTE	0087 0349 053D 0726 0DD9 13B8 173A
SCR1	054C	ABSOLUTE	0080
SCR	0547	ABSOLUTE	1CF7
SDRIV	F1D9	ABSOLUTE	05B7 05C3
SEAR1	0150	ABSOLUTE	0144
SEAR2	0155	ABSOLUTE	0148 0150 0158
SEAR	0142	ABSOLUTE	0134 014D 0461 0EBD 1781 186F
SEND	19BE	ABSOLUTE	18C9 18CE
SEN1	19C8	ABSOLUTE	19F9
SETCI	D00F	ABSOLUTE	0647
SETCO	D015	ABSOLUTE	064C
SETC	0642	ABSOLUTE	1D5D
SLAB	183D	ABSOLUTE	1461 1568 1985
SLA1	185D	ABSOLUTE	184D 1856
SLA2	1886	ABSOLUTE	185B 1882
SLA3	1889	ABSOLUTE	1863
SMTL1	139B	ABSOLUTE	139F
SMTL2	13A4	ABSOLUTE	13A6
SMTL	1399	ABSOLUTE	13B3
SOLD	06CA	ABSOLUTE	06D2
SPCHAR	F1F9	ABSOLUTE	0129 0390 0396 08CD 09AA 0BCE 101F 110C 135C 1383 15B3 15B8 15C2

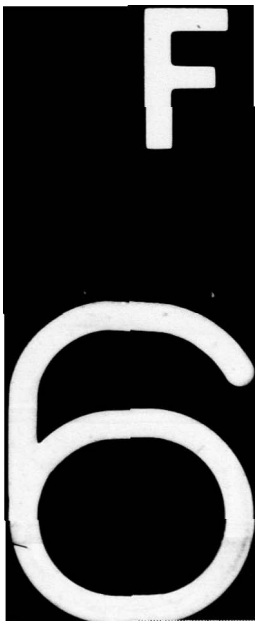
PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR	TYPE	REFERENCES
SPCHR1	F1F9	ABSOLUTE		F1FD
SPCHR2	F1FA	ABSOLUTE		08F1
SPSAV	F1D5	ABSOLUTE		00A6 0A0F 0A25 0A33 0A3F 1AC5
SRANG	F010	ABSOLUTE		099C 09D7 0A18 0AB5 0B65 0B9B
SRCHQT	1884	ABSOLUTE		1878
SRCHST	186B	ABSOLUTE		187F
SSCAND	06A0	ABSOLUTE		069B
SSCAN1	06A3	ABSOLUTE		0696
SSCAN2	06A5	ABSOLUTE		06AD
SSCAN3	06B4	ABSOLUTE		06AA
SSCAN	0694	ABSOLUTE		0670 069E 06B2
SSTRT	F012	ABSOLUTE		098E 09C9 0AE9
SSTR	193D	ABSOLUTE		1932
SSYMT	1D60	ABSOLUTE		138B
STACK	F300	ABSOLUTE		00BA 03EE 13D4 1AD9
STOM	030D	ABSOLUTE		0334 0FA0 11D8
SVSL1	0A8D	ABSOLUTE		0A87 0A98
SVSL2	0A91	ABSOLUTE		0A85 0A8B
SYMBP	1225	ABSOLUTE		1217 1267
SYMB1	1212	ABSOLUTE		1223
SYMB2	1217	ABSOLUTE		121E
SYMB	11F8	ABSOLUTE		1D21
SYMP1	1229	ABSOLUTE		1233
SYMP2	122F	ABSOLUTE		122B
SYMT	F300	ABSOLUTE		120F 138E 186B F300
SYSERR	F26E	ABSOLUTE		1360 141B 1763
S1BUF	F213	ABSOLUTE		01CE 065C
S2BUF	F23B	ABSOLUTE		01D9
TABA	F266	ABSOLUTE		1487 1512
TABE	D0A5	ABSOLUTE		0D51
TABR	D0A8	ABSOLUTE		00A9 0D56
TABST	0D4F	ABSOLUTE		1D45
TABS	D0A2	ABSOLUTE		0D59
TEMP	F27B	ABSOLUTE		17FF 181F 19C9
TYPEZ	0905	ABSOLUTE		08F6
TYPE	08E3	ABSOLUTE		0984
TYP1	164E	ABSOLUTE		167E 16D5 1714 17A7
TYP2	1651	ABSOLUTE		17B1
TYP3	1662	ABSOLUTE		17B9
TYP4	1680	ABSOLUTE		17C2
TYP5A	16C3	ABSOLUTE		16B3 16B7
TYP5	16B1	ABSOLUTE		17CA
TYP6	16F9	ABSOLUTE		17FA
TYS5	16CB	ABSOLUTE		15E9
TYS6	170B	ABSOLUTE		15A0 15D0
TY31	1673	ABSOLUTE		1659 1660 1694 1699

PROGRAM ERRORS -- 0

SYMBOL	VALUE	ADDR TYPE	REFERENCES
TY32	1676	ABSOLUTE	1696 16AF
TY41	1698	ABSOLUTE	168F
TY56	16D8	ABSOLUTE	16C5 16FD
TY6	1708	ABSOLUTE	16FB
UCOPY	D815	ABSOLUTE	0632
UDEPOS	D81B	ABSOLUTE	0623
UERR	063C	ABSOLUTE	0637
UEXAM	D818	ABSOLUTE	061E
UFIND	D81E	ABSOLUTE	062D
USPTR	D824	ABSOLUTE	0628
UTILITY	D000	ABSOLUTE	F300
UTIL	060B	ABSOLUTE	0612 1C9D
VALC	0166	ABSOLUTE	00EC
WEX	0EA8	ABSOLUTE	0E8C 0E92
WNEXT	0E6B	ABSOLUTE	0D74 0DB8
WNXE	0E94	ABSOLUTE	0E7D 0E8E 0EAD
WNXT	0E7A	ABSOLUTE	0EA1
WSBF	F004	ABSOLUTE	0DBB 0DC8 0DE7 0DEF 0DFD 0E6F 0E87 0EA3 0EAA 0F29 0F38 0F47 0F54
WSOK	0DE3	ABSOLUTE	0DD3
WSPE	F002	ABSOLUTE	008C 0DCD 136B
WSPS	F000	ABSOLUTE	0086 0DFA 0E6B
XARIAN	0000	ABSOLUTE	
XBRKP	0008	ABSOLUTE	
XCKA11	0040	ABSOLUTE	
XCKA21	0043	ABSOLUTE	
XCKFN1	0046	ABSOLUTE	
XDCOM	0069	ABSOLUTE	
XDSCAN	0049	ABSOLUTE	
XEOR	0066	ABSOLUTE	1D71
XFILE	004C	ABSOLUTE	
XFIND	004F	ABSOLUTE	
XFSEA	005E	ABSOLUTE	
XINK	0072	ABSOLUTE	1D7D
XLINE	0052	ABSOLUTE	
XLOAD	0055	ABSOLUTE	
XMESS	005B	ABSOLUTE	
XPARSE	006C	ABSOLUTE	1D95
XREAD	006F	ABSOLUTE	1D77
XRESET	0004	ABSOLUTE	
XRNUM	0061	ABSOLUTE	
XSAVE	0058	ABSOLUTE	
XTRAP	0038	ABSOLUTE	
ZBUF1	0161	ABSOLUTE	0163
ZBUF	015B	ABSOLUTE	144A 149B 1553 18BF
ZERO	0FF0	ABSOLUTE	0FCA 0FDE

50473 MILLISECONDS USED IN THIS ASSEMBLY
THIS ASSEMBLY MADE ON 01/04/80 AT 21.04.22.


F
6